KaHyPar

KARLSRUHE HYPERGRAPH PARTITIONING

Modern Hypergraph Partitioning: KaHyPar, Mt-KaHyPar, and Beyond

ISPD 2026 · March, 18

Sebastian Schlag, **Tobias Heuer**[1], **Christian Schulz**[2], Yaroslav Akhremtsev[3], Robin Andre[3], Adil Chhabra[2], Lars Gottesbüren[3], Michael Hamann[3],Vitali Henne[3], Nikolai Maas[3], Henning Meyerhenke[3], Peter Sanders[3], Daniel Seemaier[3], Bora Uçar[4], Dorothea Wagner[3], Loris Wilwert[2]

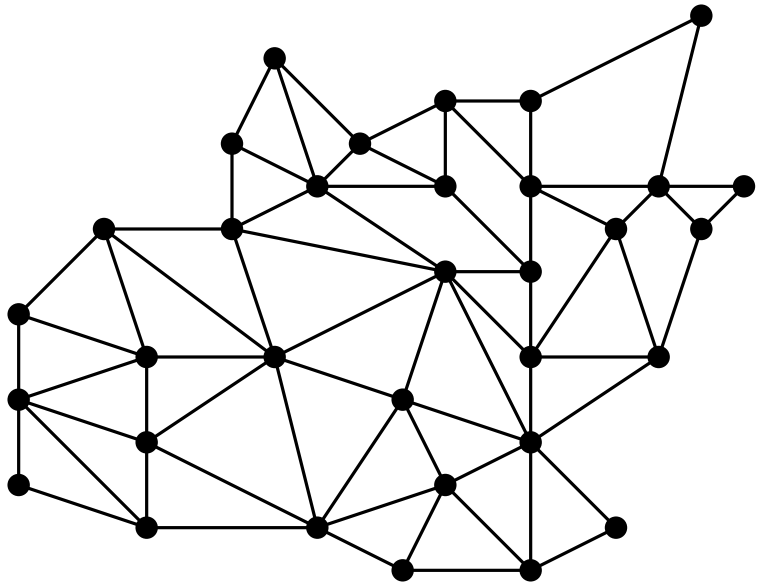
[1]eBay Inc., [2]Heidelberg University, [3] Karlsruhe Institute Of Technology,
[4] CNRS and LIP (UMR 5668 CNRS - ENS Lyon - UCB Lyon 1 - Inria)

Graphs and Hypergraphs

Graph $G = (V, E)$

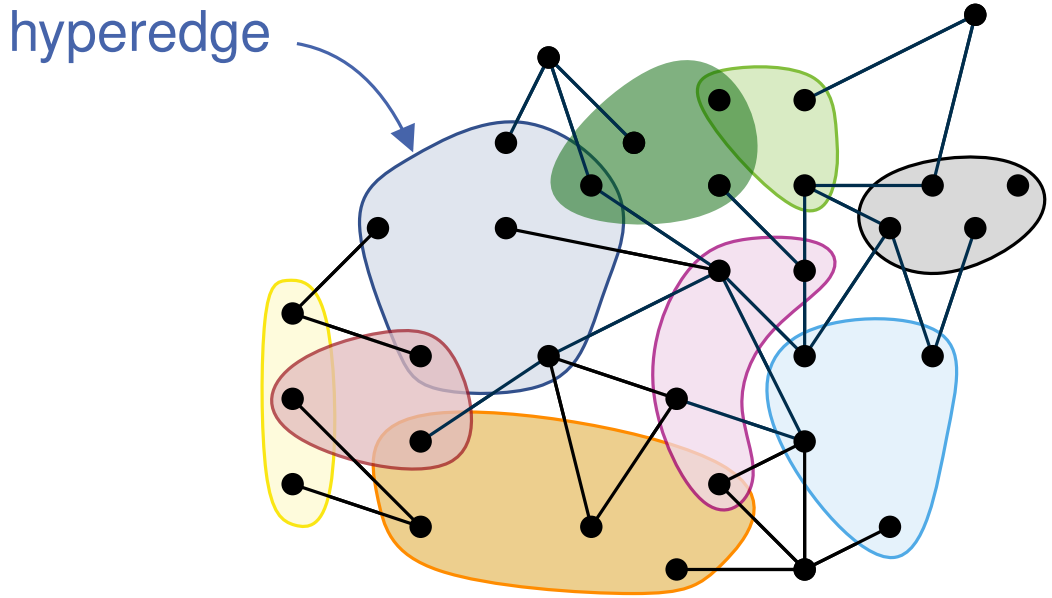
vertices — edges

- Models relationships between objects
- Dyadic (2-ary) relationships



Hypergraph $H = (V, E)$

- Generalization of a graph
  $\Rightarrow$ hyperedges connect ≥ 2 vertices
- Arbitrary ($\mathbf{d}$-ary) relationships
- Edge set $E \subseteq \mathcal{P}(V) \setminus \emptyset$



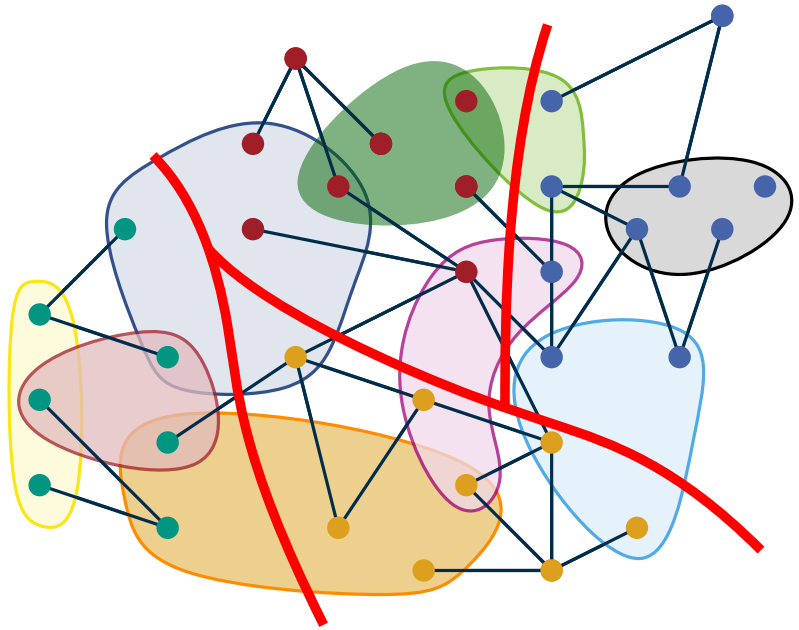
ε-Balanced Hypergraph Partitioning (HGP)

Partition hypergraph $H = (V, E, c : V \to \mathrm{R}_{>0}, \omega : E \to \mathrm{R}_{>0})$ into **k** disjoint blocks $\Pi = \{V_1, \ldots, V_k\}$ such that

- Blocks V_i are roughly equal-sized:

$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$

- Objective function on hyperedges is minimized



ε-Balanced Hypergraph Partitioning (HGP)

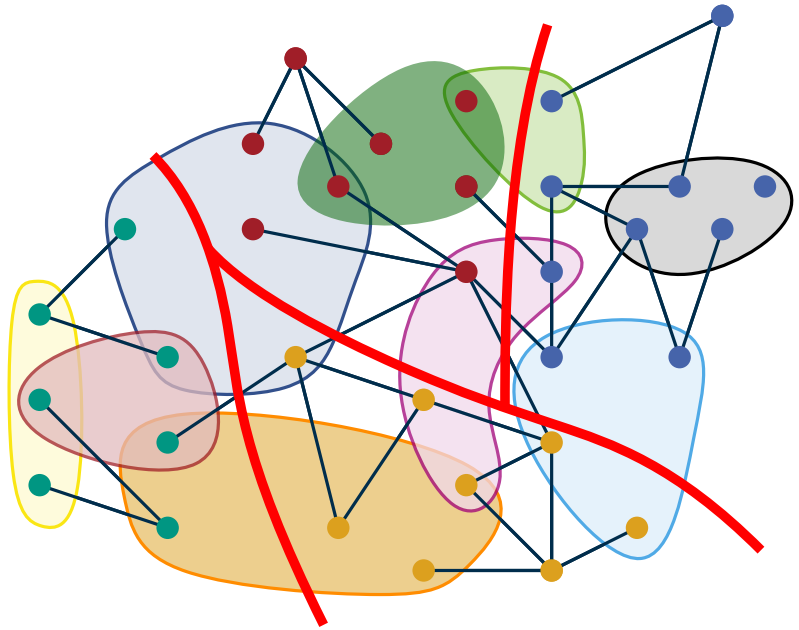
Partition hypergraph $H = (V, E, c : V \to \mathrm{R}_{>0}, \omega : E \to \mathrm{R}_{>0})$ into **k** disjoint blocks $\Pi = \{V_1, \ldots, V_k\}$ such that

- Blocks V_i are roughly equal-sized:

imbalance parameter

$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$

- Objective function on hyperedges is minimized



ε-Balanced Hypergraph Partitioning (HGP)

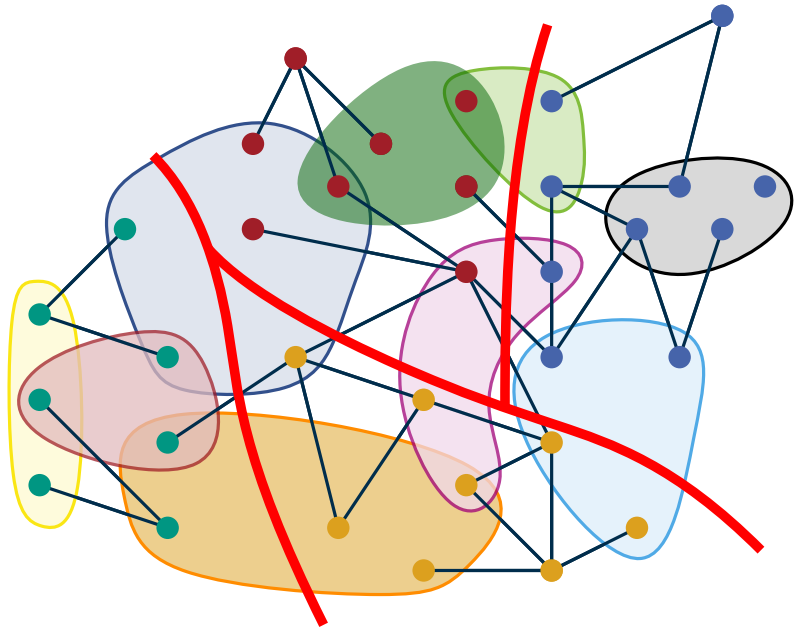
Partition hypergraph $H = (V, E, c : V \to \mathrm{R}_{>0}, \omega : E \to \mathrm{R}_{>0})$ into **k** disjoint blocks $\Pi = \{V_1, \ldots, V_k\}$ such that

- Blocks V_i are roughly equal-sized:

imbalance parameter

$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$

- Objective function on hyperedges is minimized



Common HGP Objectives:

- Cut-Net: $\sum_{e \in \mathrm{Cut}} \omega(e)$

ε-Balanced Hypergraph Partitioning (HGP)

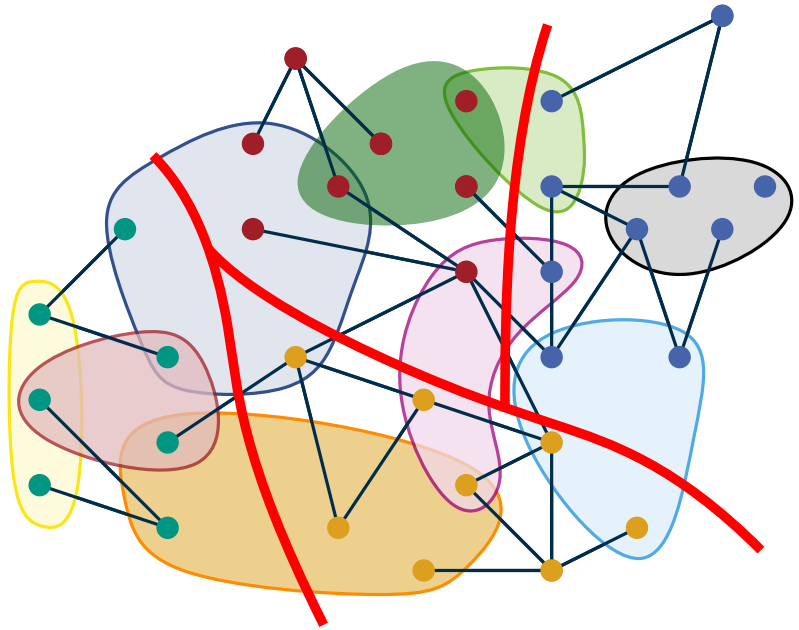
Partition hypergraph $H = (V, E, c : V \to \mathrm{R}_{>0}, \omega : E \to \mathrm{R}_{>0})$ into **k** disjoint blocks $\Pi = \{V_1, \ldots, V_k\}$ such that

- Blocks V_i are roughly equal-sized:

imbalance parameter

$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$

- Objective function on hyperedges is minimized



Common HGP Objectives:

- Cut-Net: $\sum_{e \in \mathrm{Cut}} \omega(e)$
- Connectivity: $\sum_{e \in \mathrm{cut}} (\lambda - 1)\, \omega(e)$

ε-Balanced Hypergraph Partitioning (HGP)

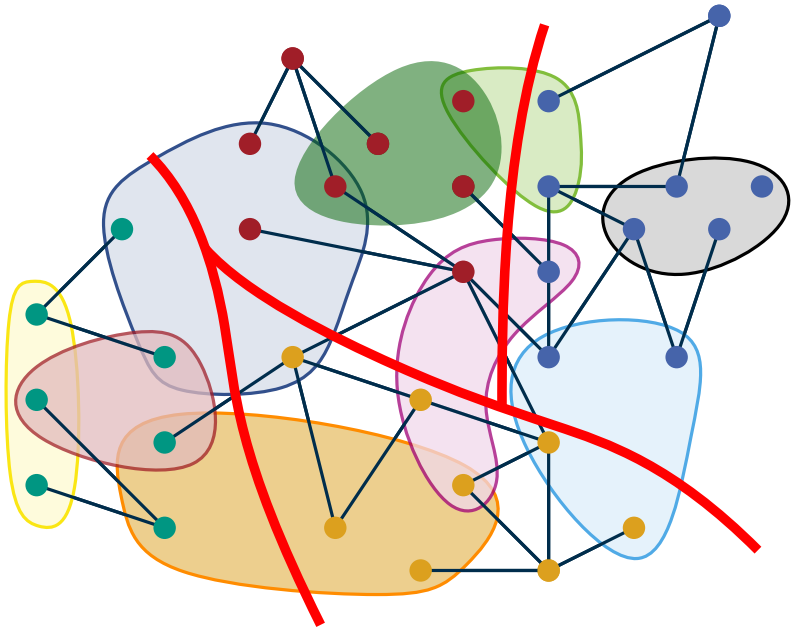
Partition hypergraph $H = (V, E, c : V \to \mathrm{R}_{>0}, \omega : E \to \mathrm{R}_{>0})$ into **k** disjoint blocks $\Pi = \{V_1, \ldots, V_k\}$ such that

- Blocks V_i are roughly equal-sized:

imbalance parameter

$$c(V_i) \leq (1 + \varepsilon) \left\lceil \frac{c(V)}{k} \right\rceil$$

- Objective function on hyperedges is minimized



Common HGP Objectives:

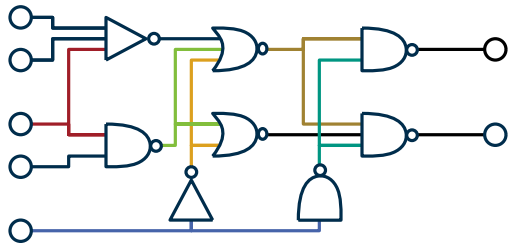
- Cut-Net: $\sum_{e \in \mathrm{Cut}} \omega(e)$
- Connectivity: $\sum_{e \in \mathrm{cut}} (\lambda - 1)\, \omega(e)$

 # blocks connected by e

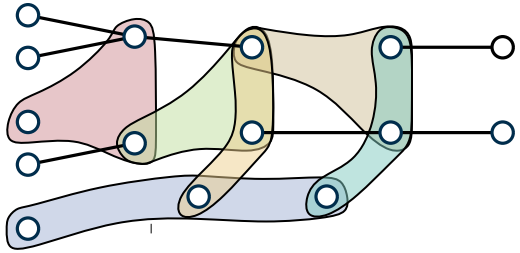
Well-Known HGP Applications

Circuit Design

Application



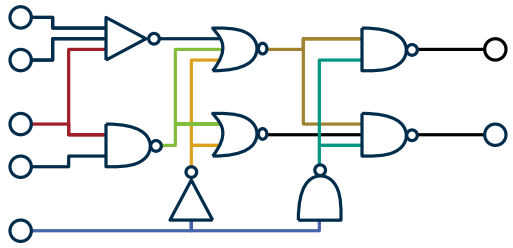
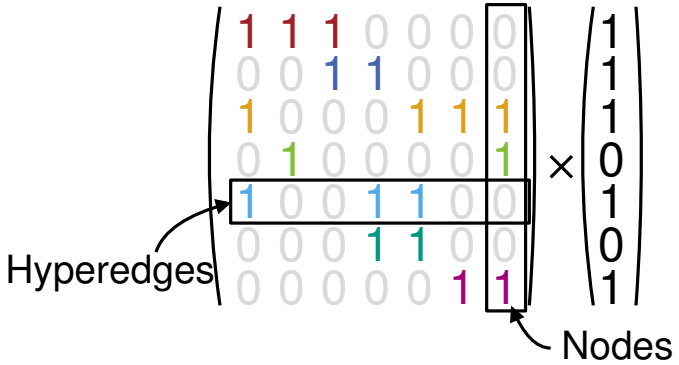
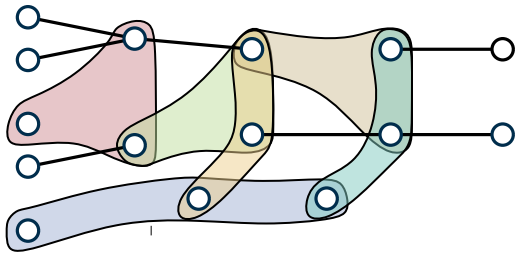
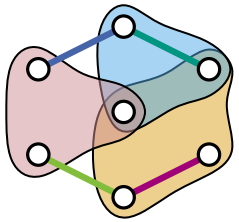
Model



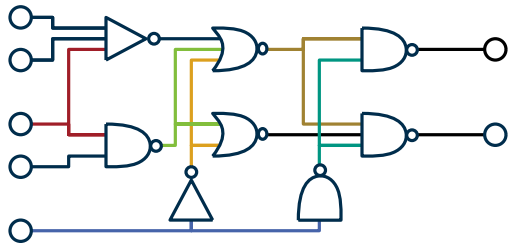
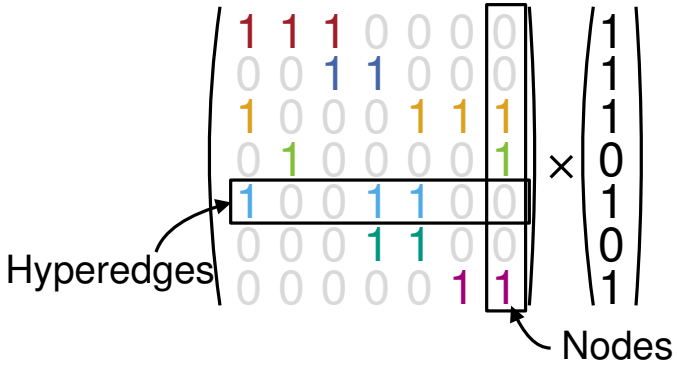
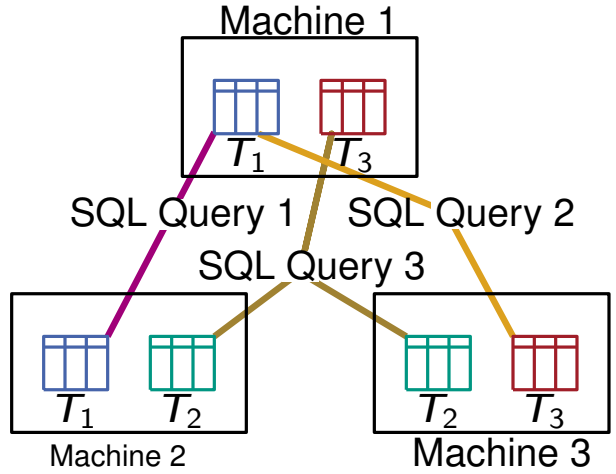
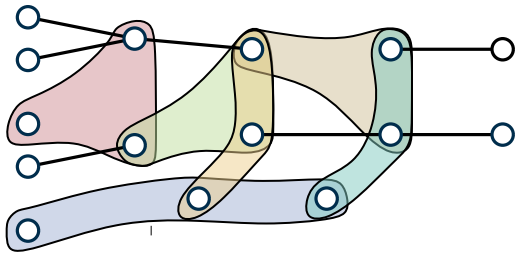
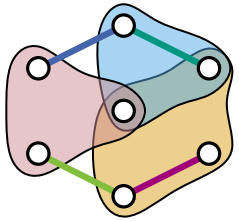
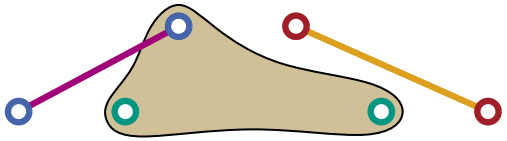
Goal

Facilitate Placement

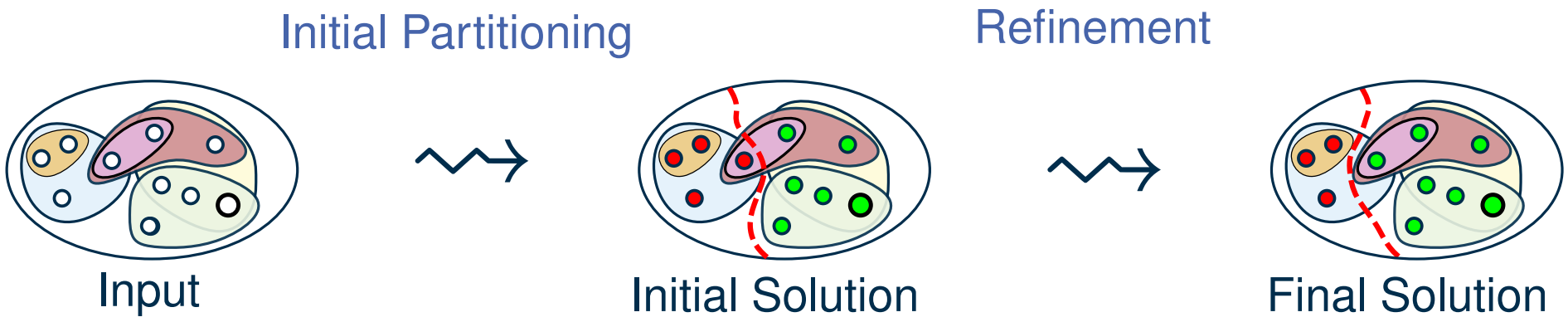
Well-Known HGP Applications

	Circuit Design	Scientific Computing
Application		
Model		
Goal	Facilitate Placement	Minimize Communication

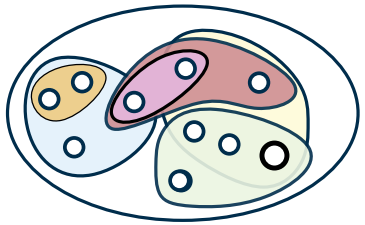
Well-Known HGP Applications

	Circuit Design	Scientific Computing	Distributed Databases
Application			
Model			
Goal	Facilitate Placement	Minimize Communication	Minimize distr. Queries

Classification: Single-Level Algorithms

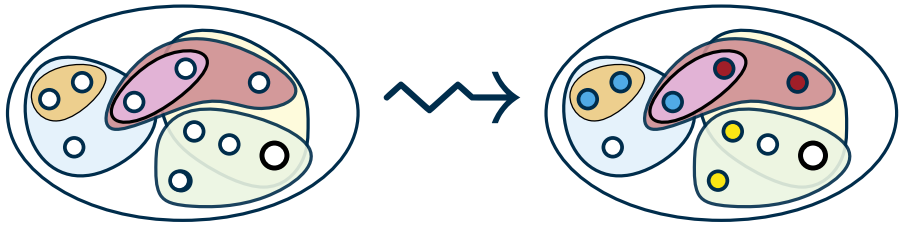


Classification: Two-Level Algorithms

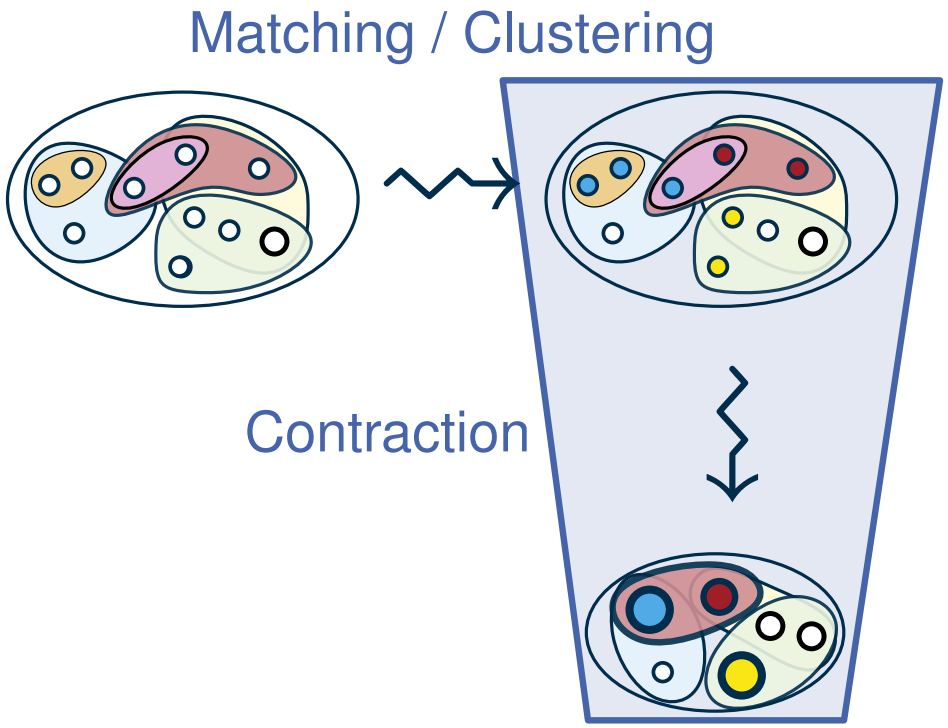


Classification: Two-Level Algorithms

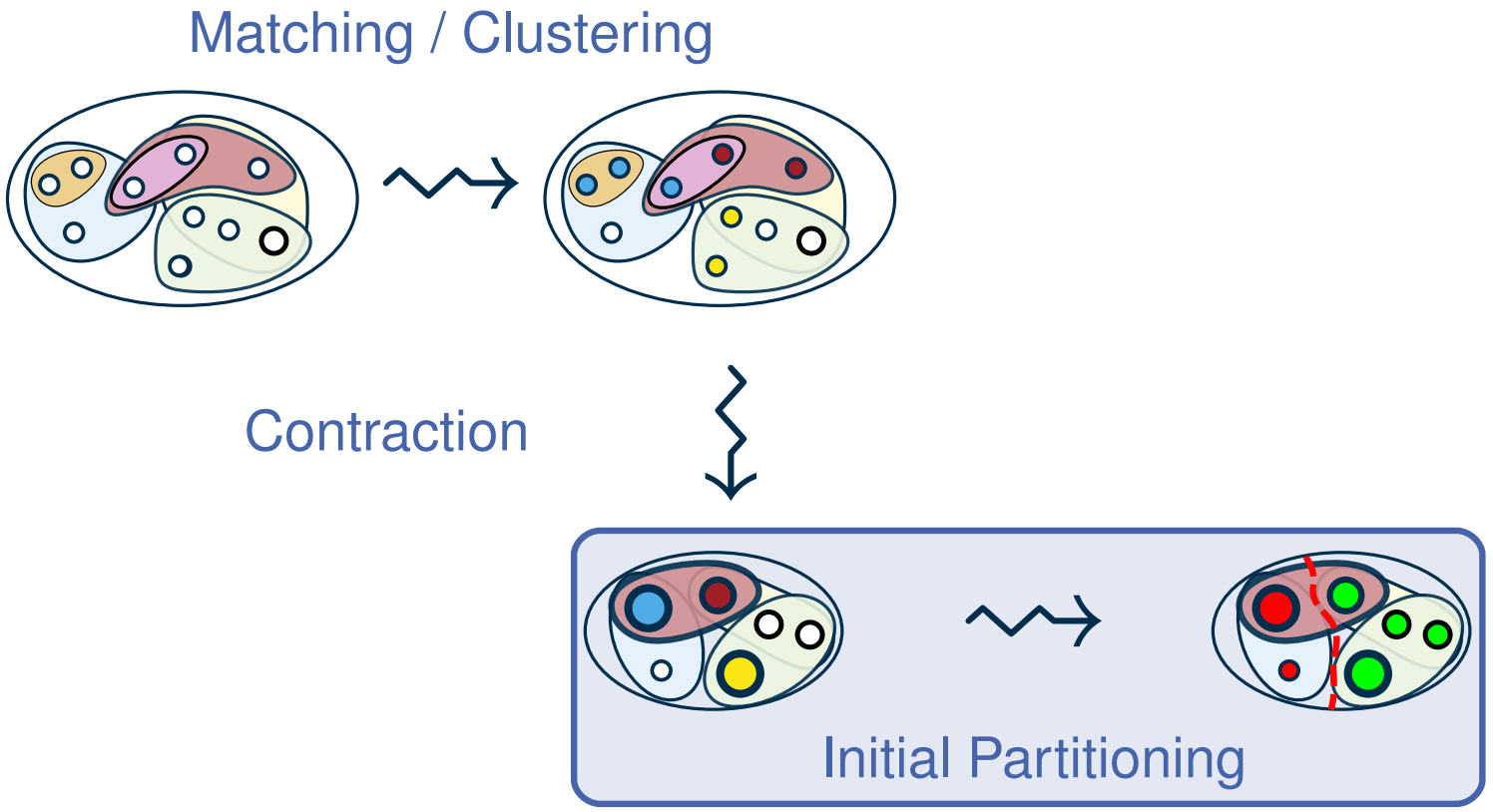
Matching / Clustering



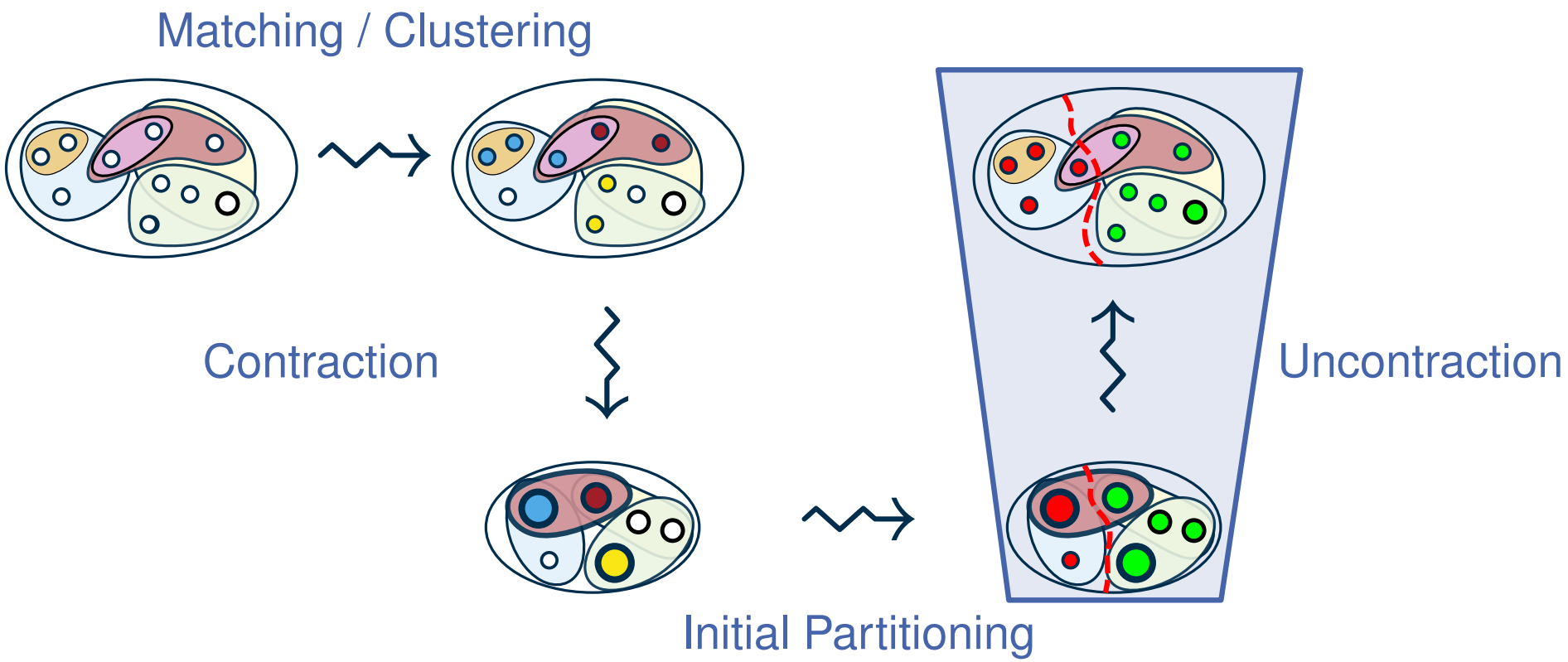
Classification: Two-Level Algorithms



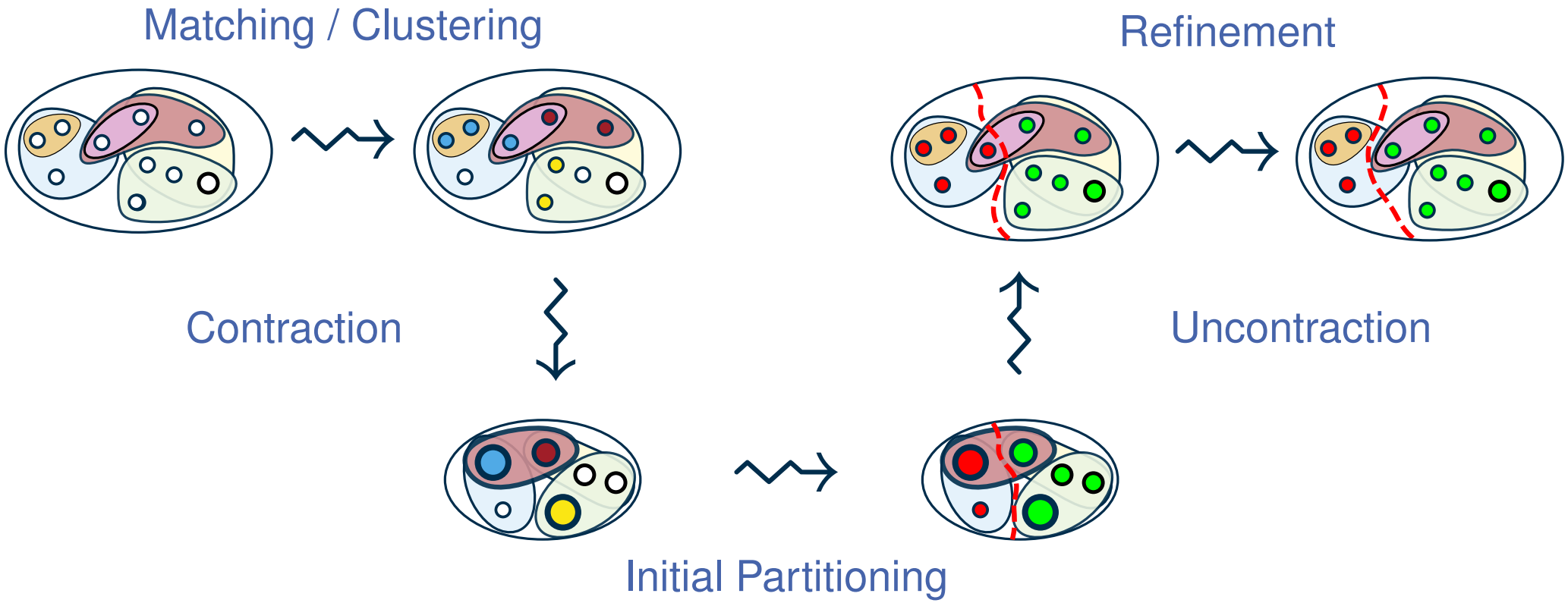
Classification: Two-Level Algorithms



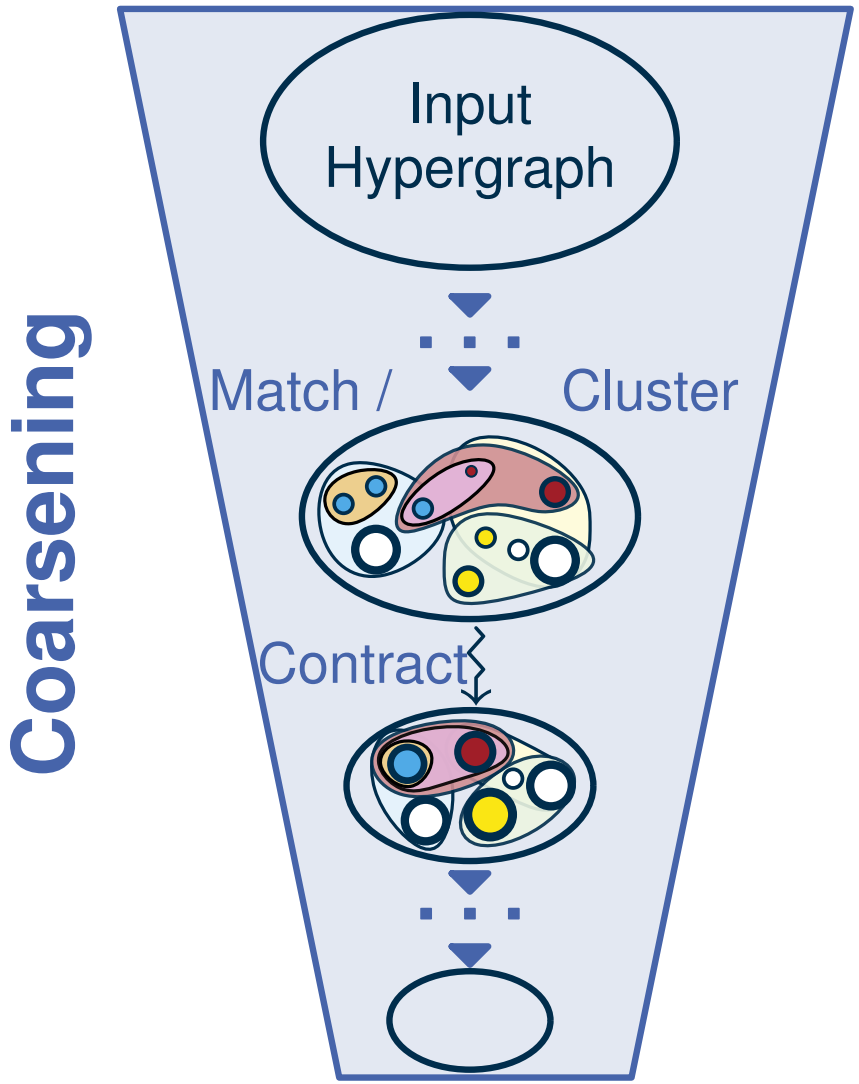
Classification: Two-Level Algorithms



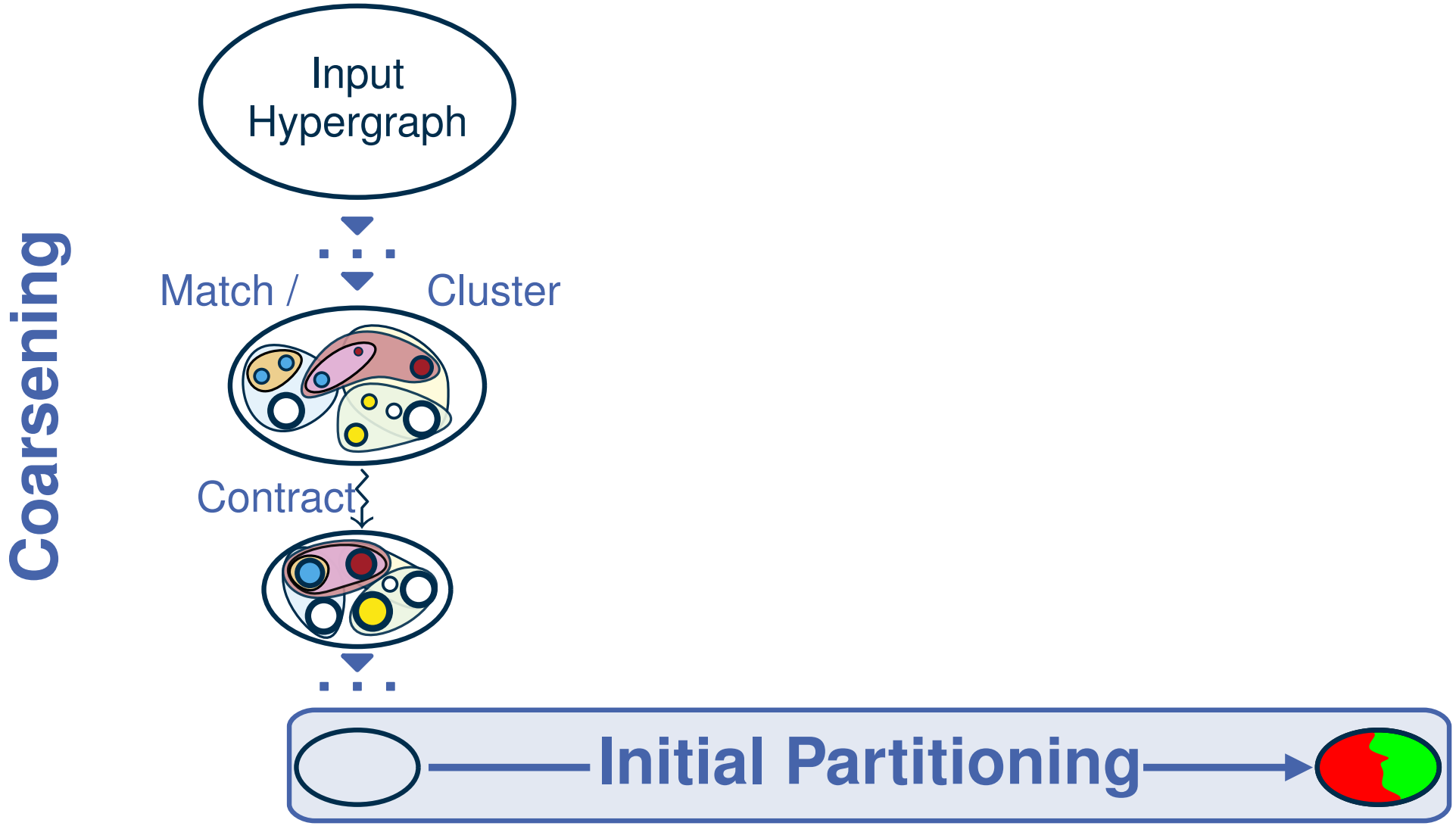
Classification: Two-Level Algorithms



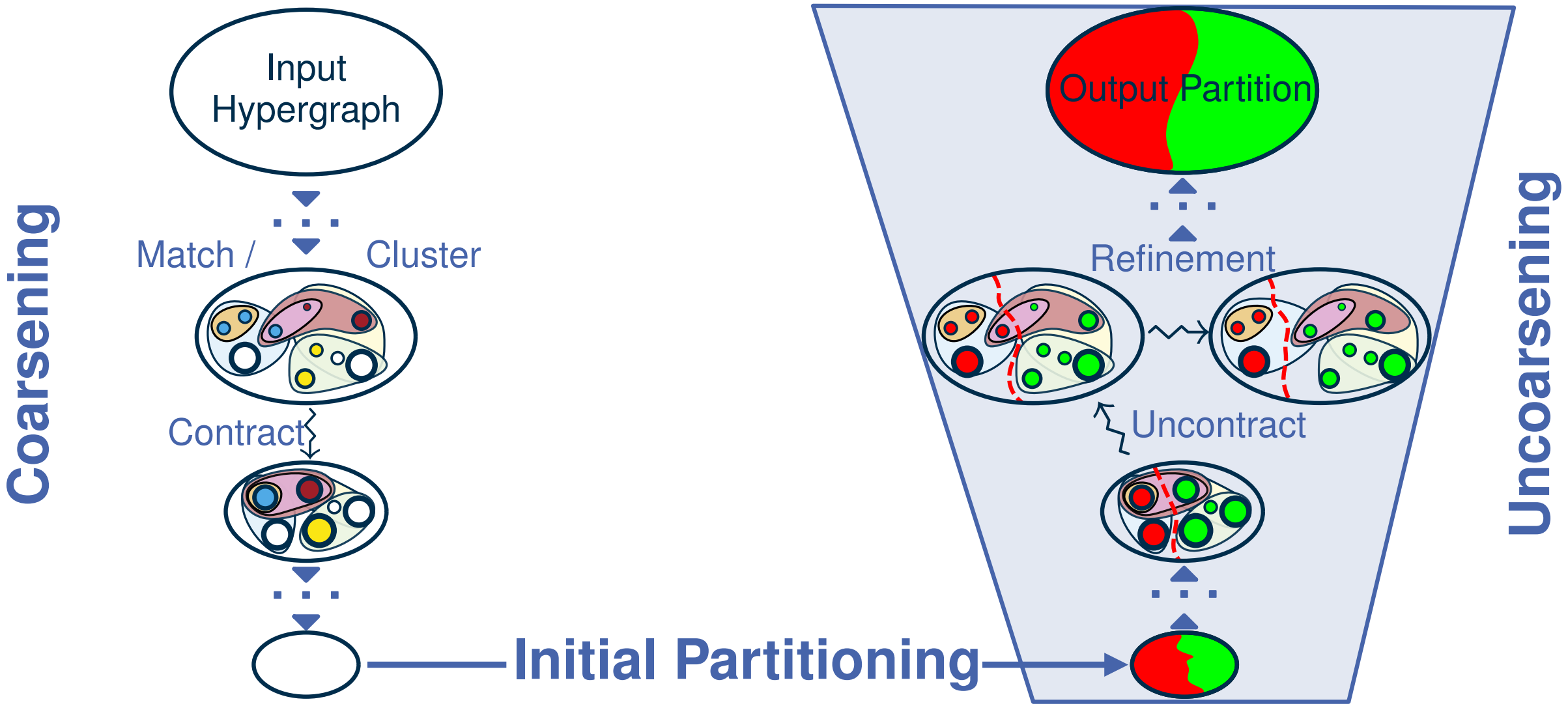
Classification: Multi-Level Algorithms



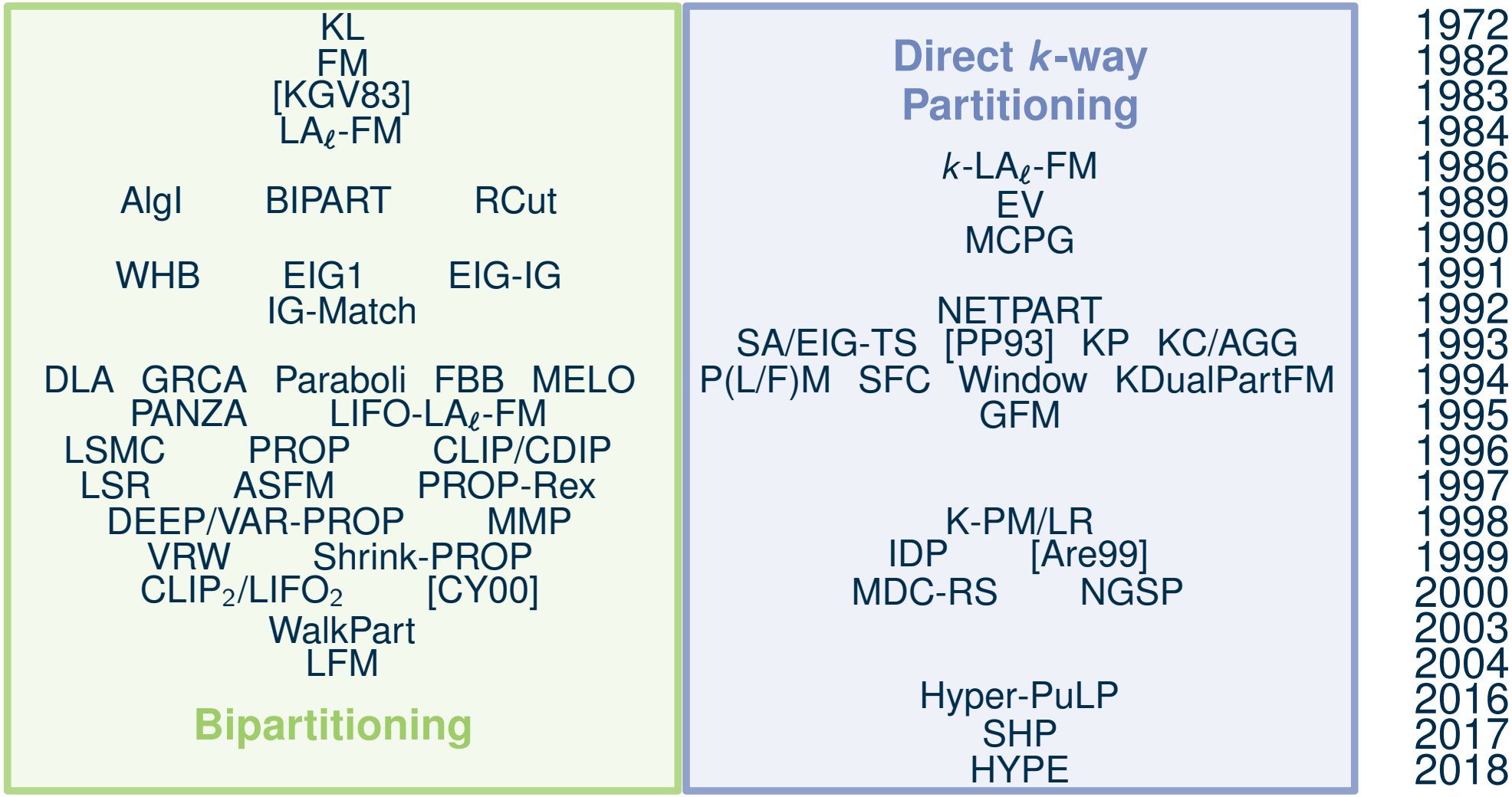
Classification: Multi-Level Algorithms



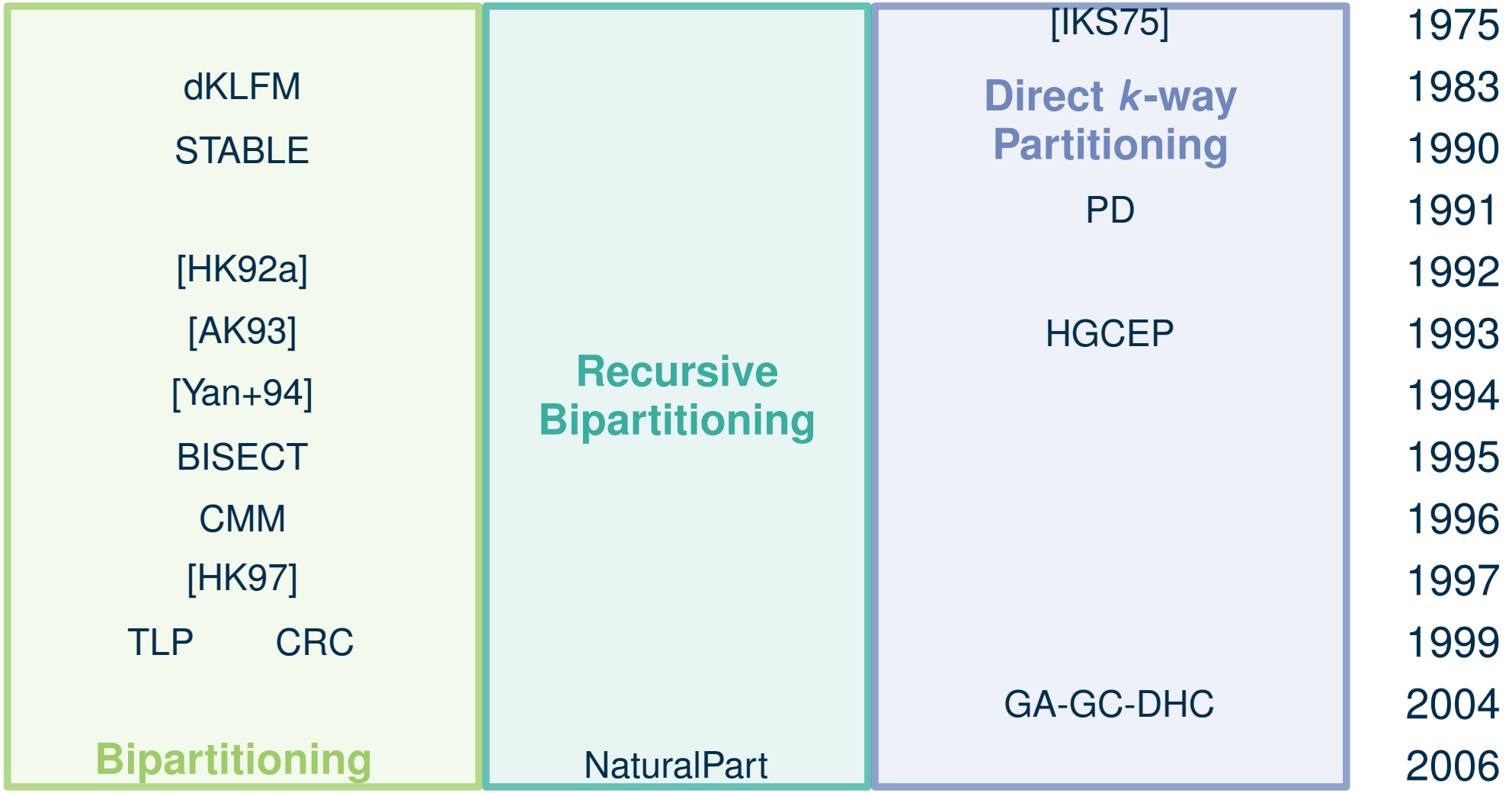
Classification: Multi-Level Algorithms



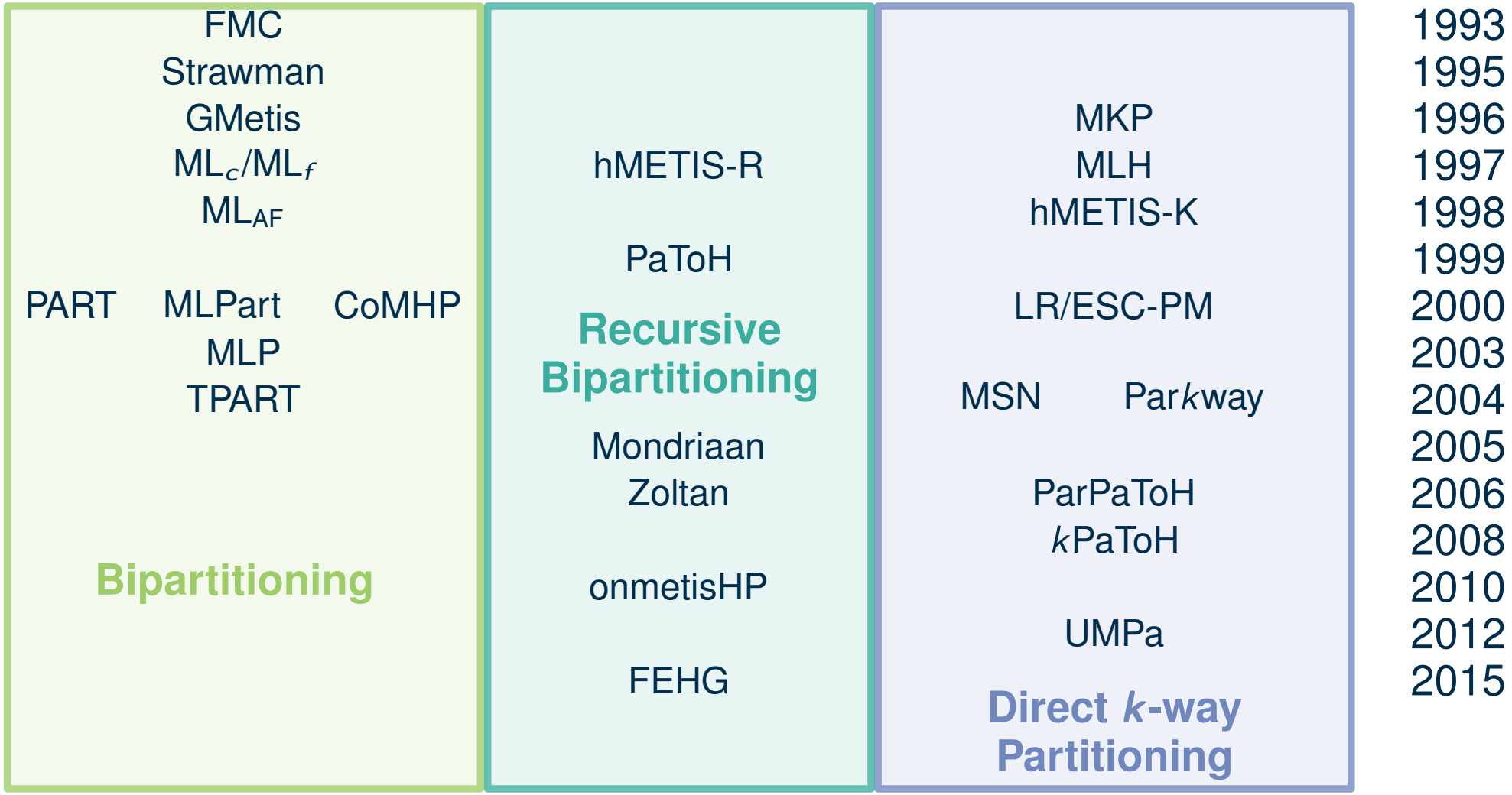
Prior Work: Single-Level Algorithms



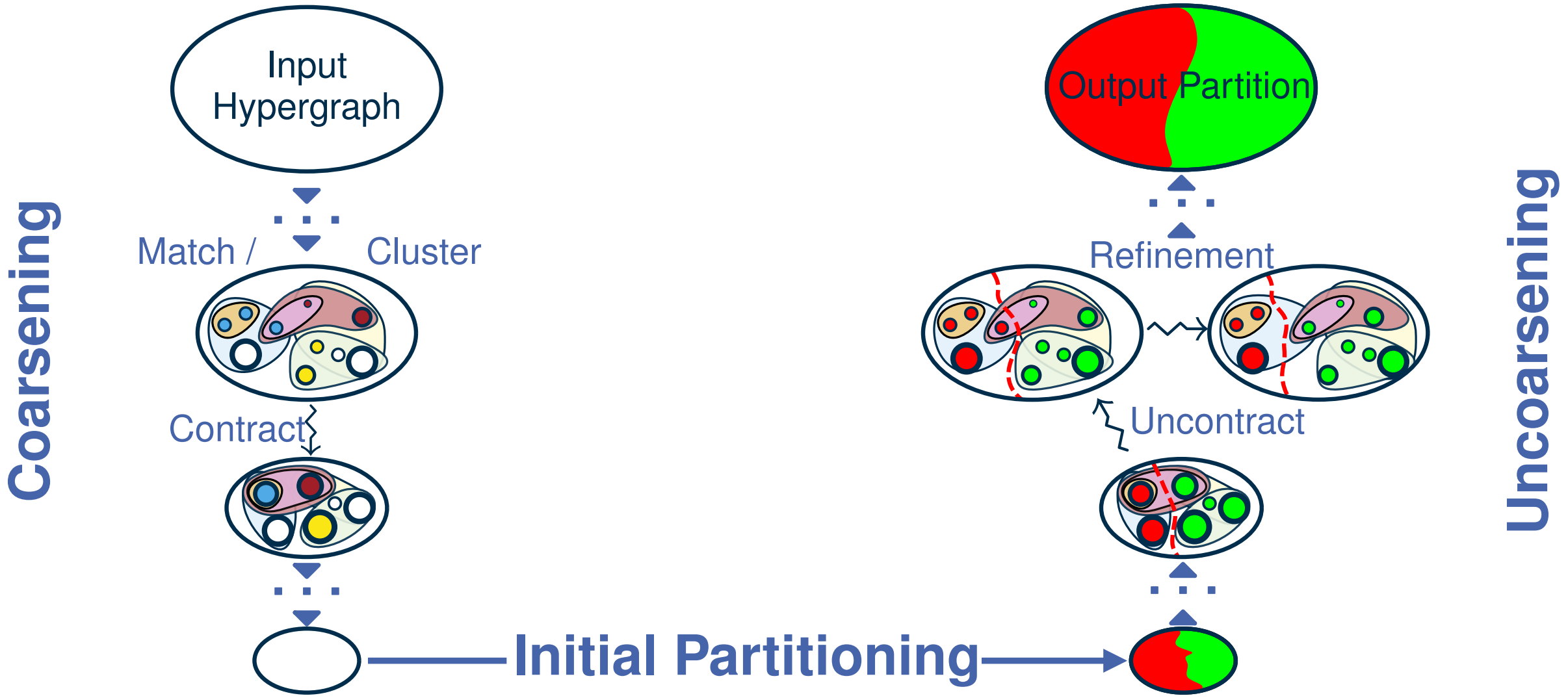
Prior Work: Two-Level Algorithms



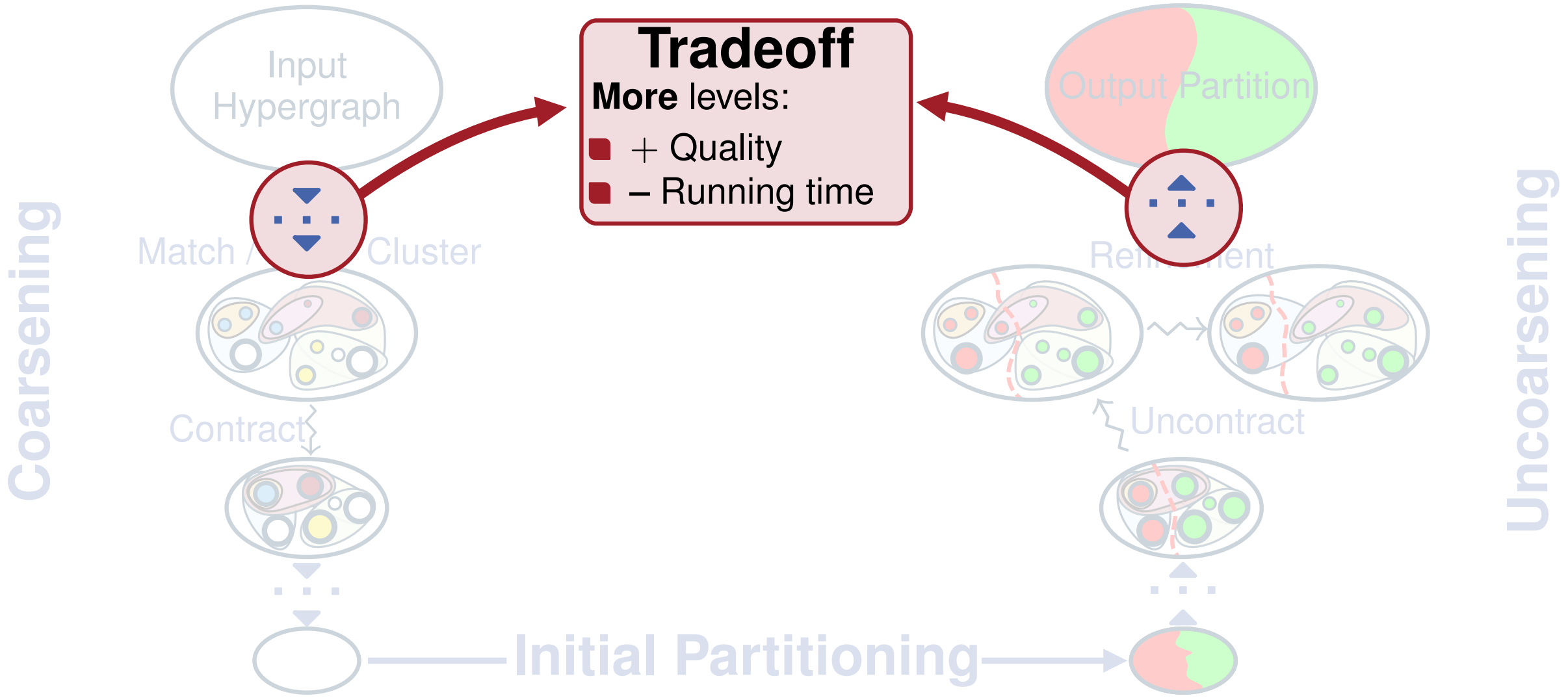
Prior Work: Multi-Level Algorithms



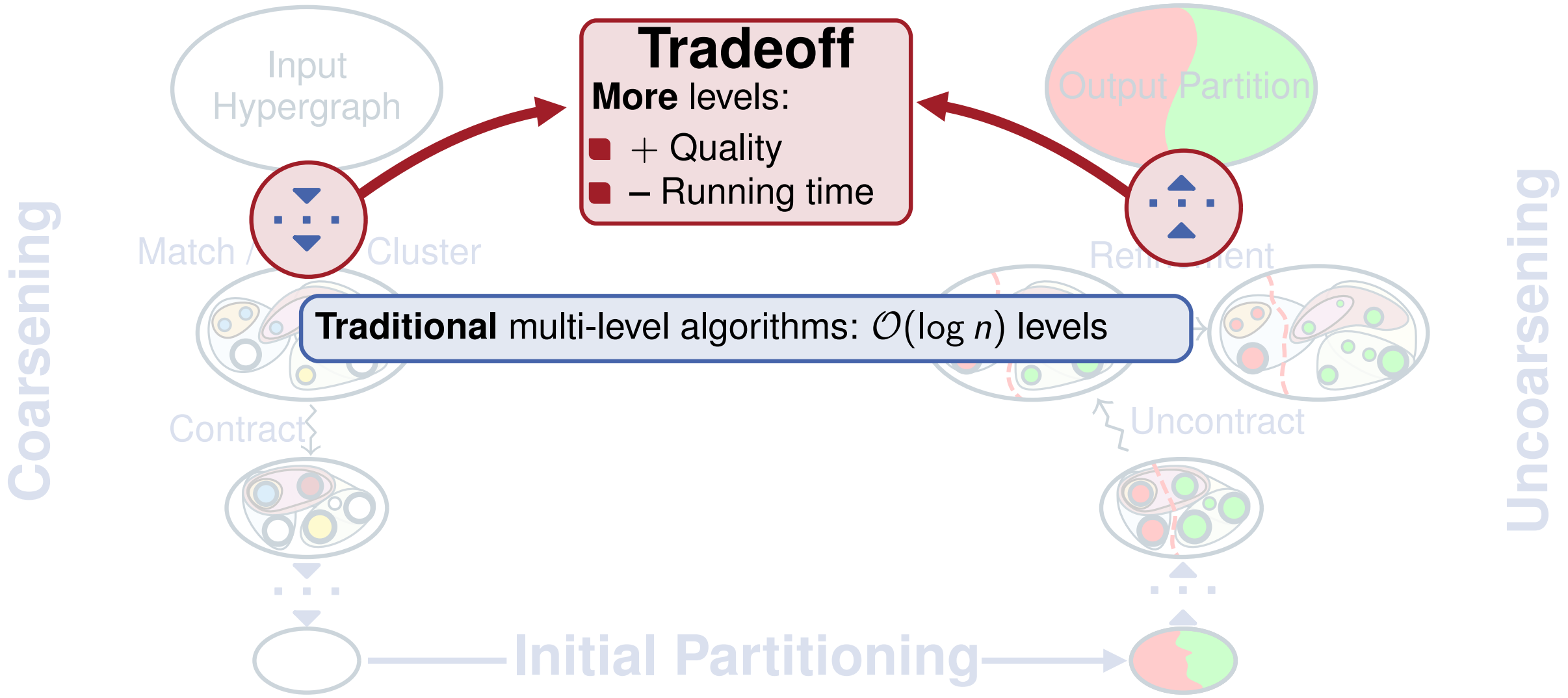
Why Yet Another Multi-Level Algorithm?



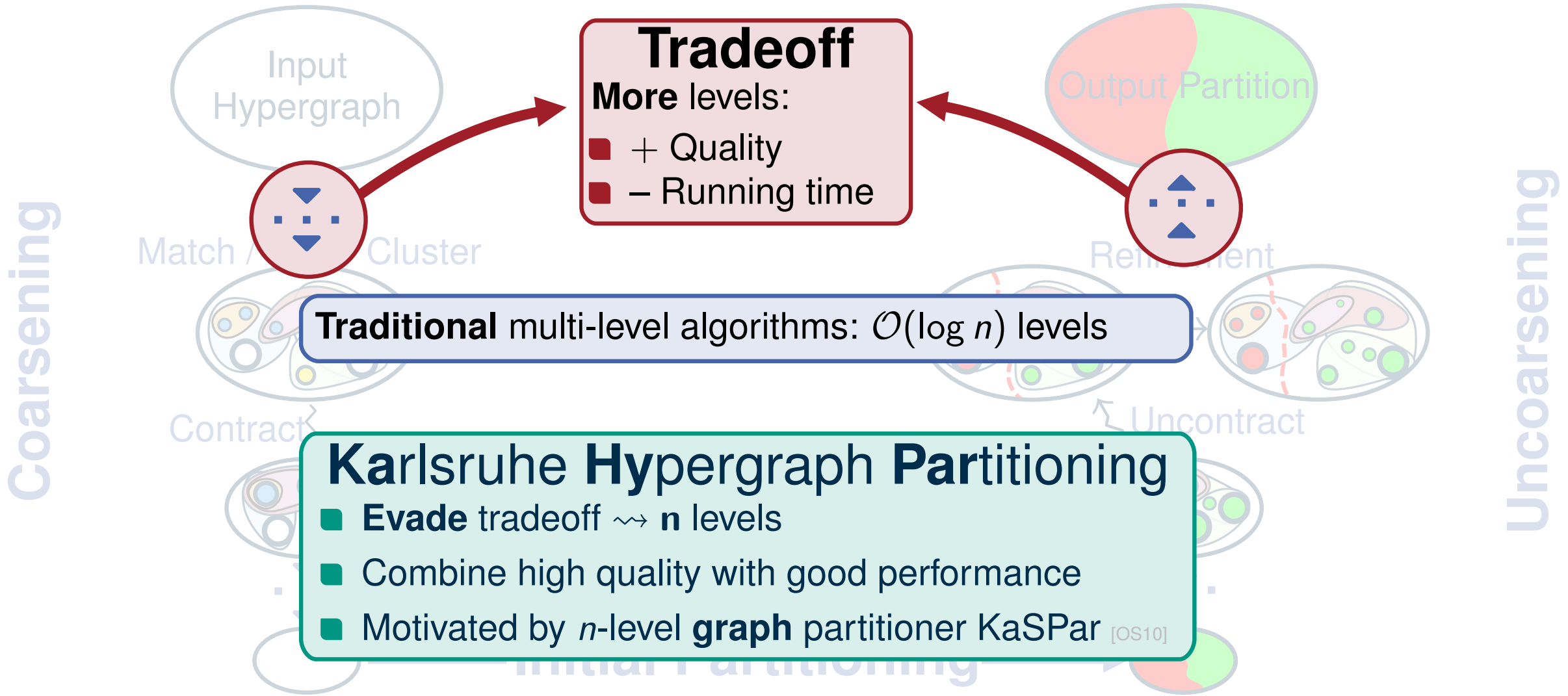
Why Yet Another Multi-Level Algorithm?



Why Yet Another Multi-Level Algorithm?

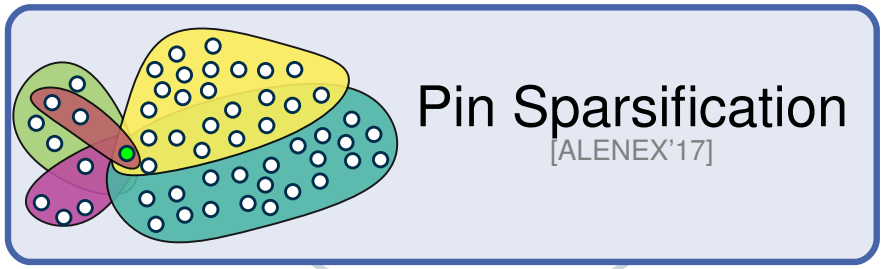


Why Yet Another Multi-Level Algorithm?



Overview: Main Algorithmic Ingredients

Overview: Main Algorithmic Ingredients



Pin Sparsification
[ALENEX'17]

Coarsening

Match / Cluster

Contract

Initial Partitioning

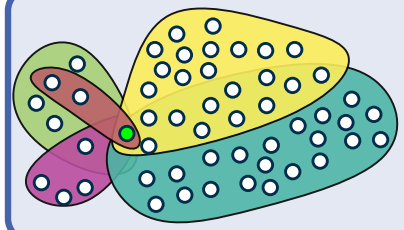
Refinement

Uncontract

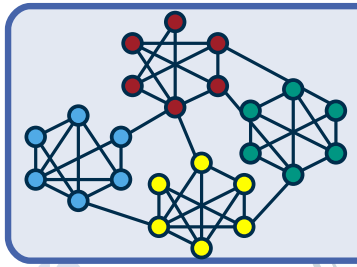
Output Partition

Uncoarsening

Overview: Main Algorithmic Ingredients

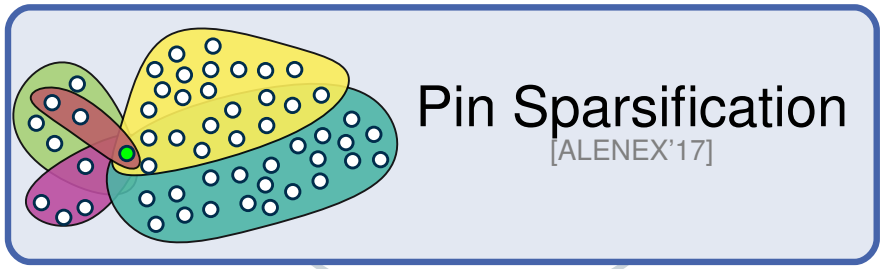


Pin Sparsification
[ALENEX'17]

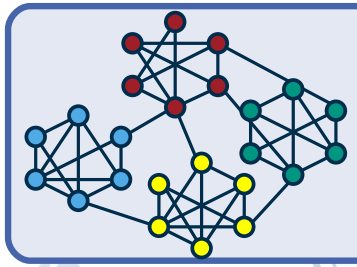


Community-Aware Coarsening
[SEA'17]

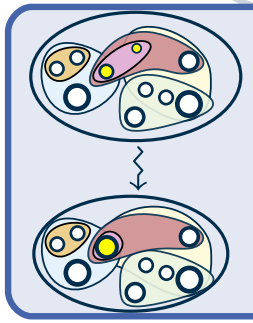
Overview: Main Algorithmic Ingredients



Pin Sparsification
[ALENEX'17]

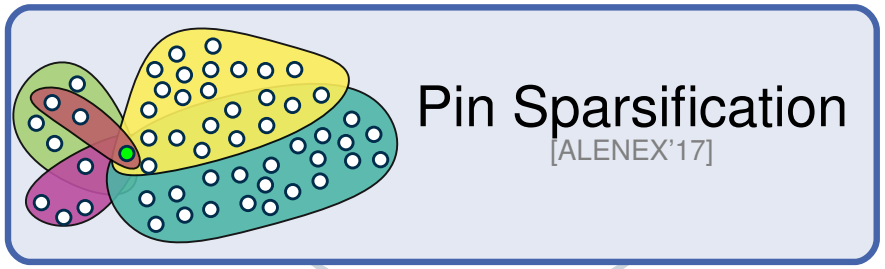


Community-Aware Coarsening
[SEA'17]

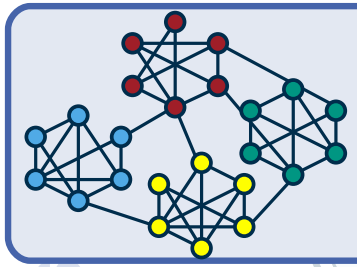


Fast *n*-Level Coarsening
[ALENEX'16]
[ALENEX'17]

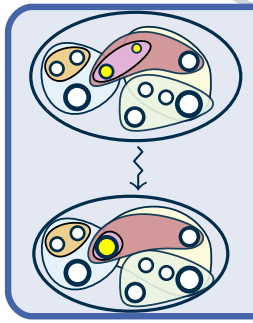
Overview: Main Algorithmic Ingredients



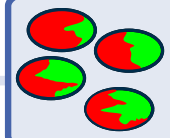
Pin Sparsification
[ALENEX'17]



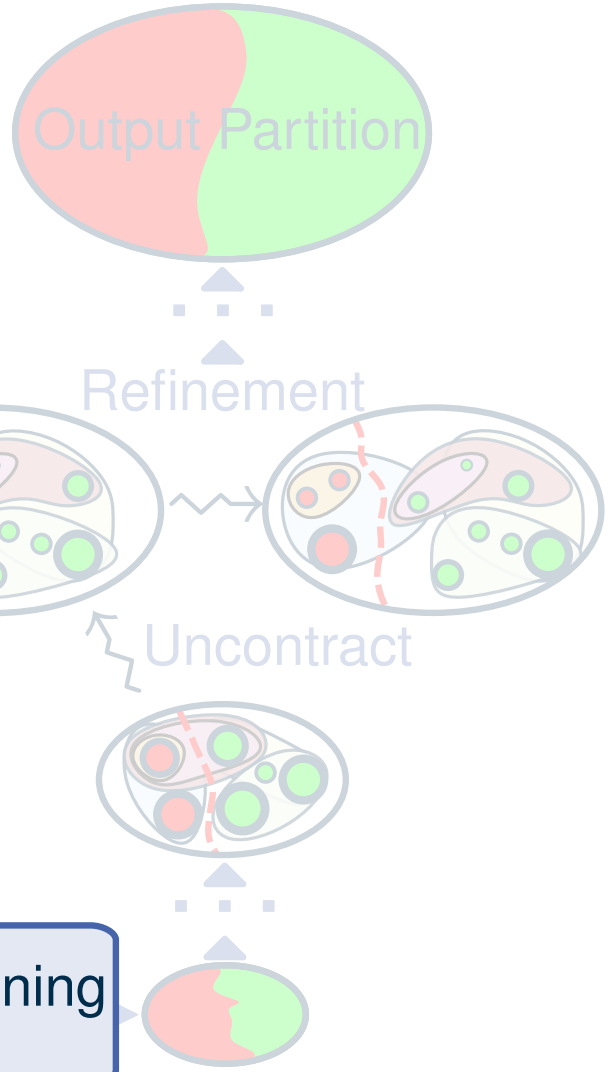
Community-Aware Coarsening
[SEA'17]



Fast *n*-Level Coarsening
[ALENEX'16]
[ALENEX'17]

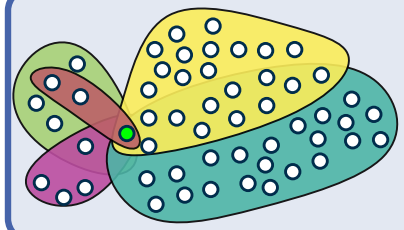


Portfolio-based Initial Partitioning
[ALENEX'16]

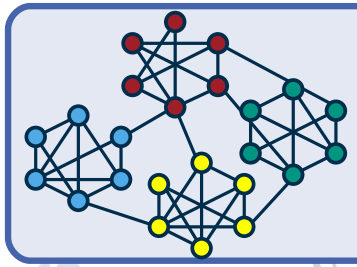


Uncoarsening

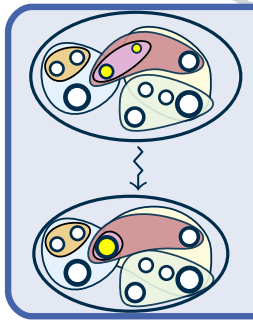
Overview: Main Algorithmic Ingredients



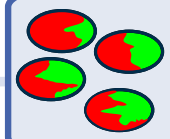
Pin Sparsification
[ALENEX'17]



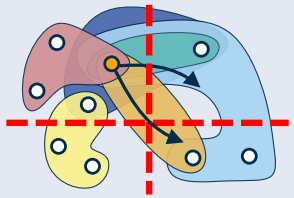
Community-Aware Coarsening
[SEA'17]



Fast *n*-Level Coarsening
[ALENEX'16]
[ALENEX'17]

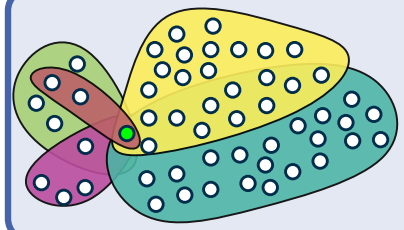


Portfolio-based Initial Partitioning
[ALENEX'16]

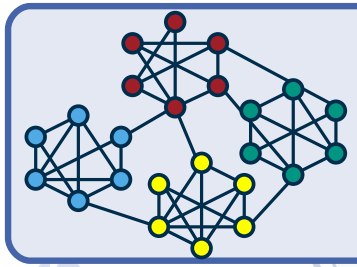


Engineered 2-way & *k*-way localized FM Algorithms
[ALENEX'16] [ALENEX'17]

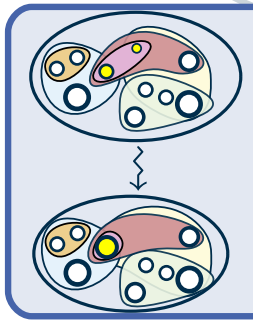
Overview: Main Algorithmic Ingredients



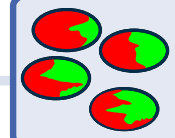
Pin Sparsification
[ALENEX'17]



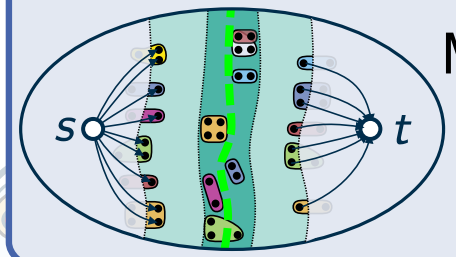
Community-Aware Coarsening
[SEA'17]



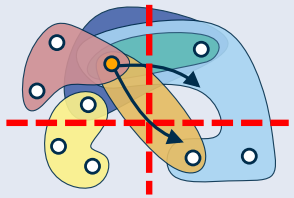
Fast *n*-Level Coarsening
[ALENEX'16]
[ALENEX'17]



Portfolio-based Initial Partitioning
[ALENEX'16]

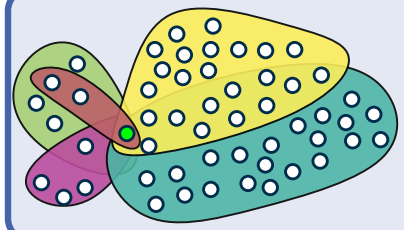


Max-Flow Min-Cut Refinement
[SEA'18] [JEA'19] [SEA'20]

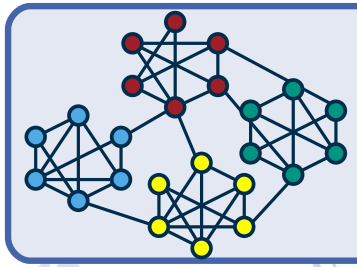


Engineered 2-way & *k*-way localized FM Algorithms
[ALENEX'16] [ALENEX'17]

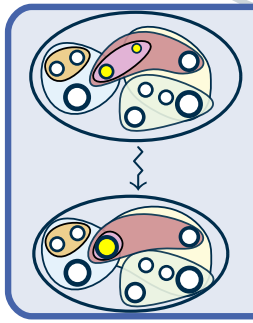
Overview: Main Algorithmic Ingredients



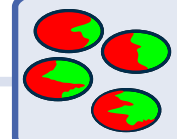
Pin Sparsification
[ALENEX'17]



Community-Aware Coarsening
[SEA'17]

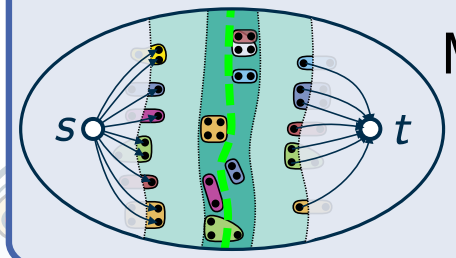


Fast *n*-Level Coarsening
[ALENEX'16]
[ALENEX'17]

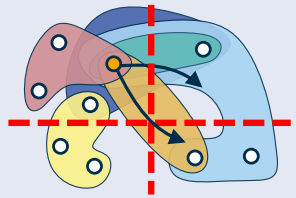


Portfolio-based Initial Partitioning
[ALENEX'16]

Memetic Multi-Level Algorithm
[GECCO'18]

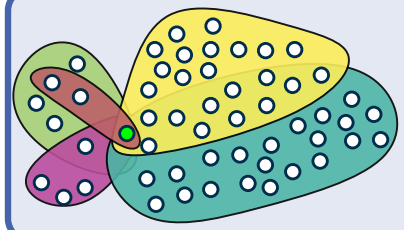


Max-Flow Min-Cut Refinement
[SEA'18] [JEA'19] [SEA'20]

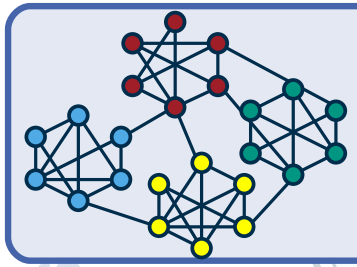


Engineered 2-way & *k*-way localized FM Algorithms
[ALENEX'16] [ALENEX'17]

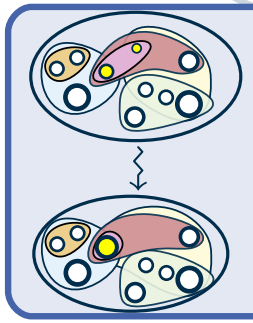
Overview: Main Algorithmic Ingredients



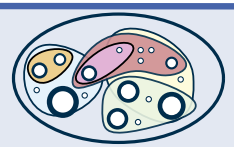
Pin Sparsification
[ALENEX'17]



Community-Aware Coarsening
[SEA'17]

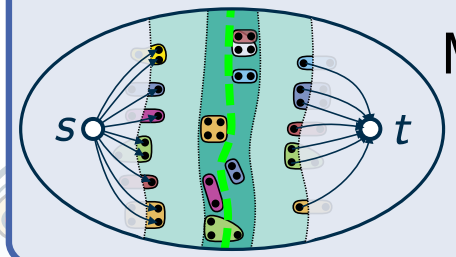


Fast n-Level Coarsening
[ALENEX'16]
[ALENEX'17]

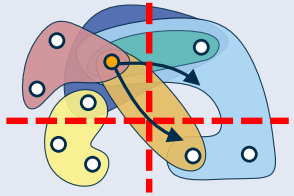


Deep Balance for weighted Hypergraphs
[SEA'21]

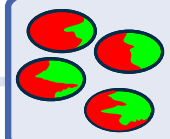
Memetic Multi-Level Algorithm
[GECCO'18]



Max-Flow Min-Cut Refinement
[SEA'18] [JEA'19] [SEA'20]



Engineered 2-way & k-way localized FM Algorithms
[ALENEX'16] [ALENEX'17]

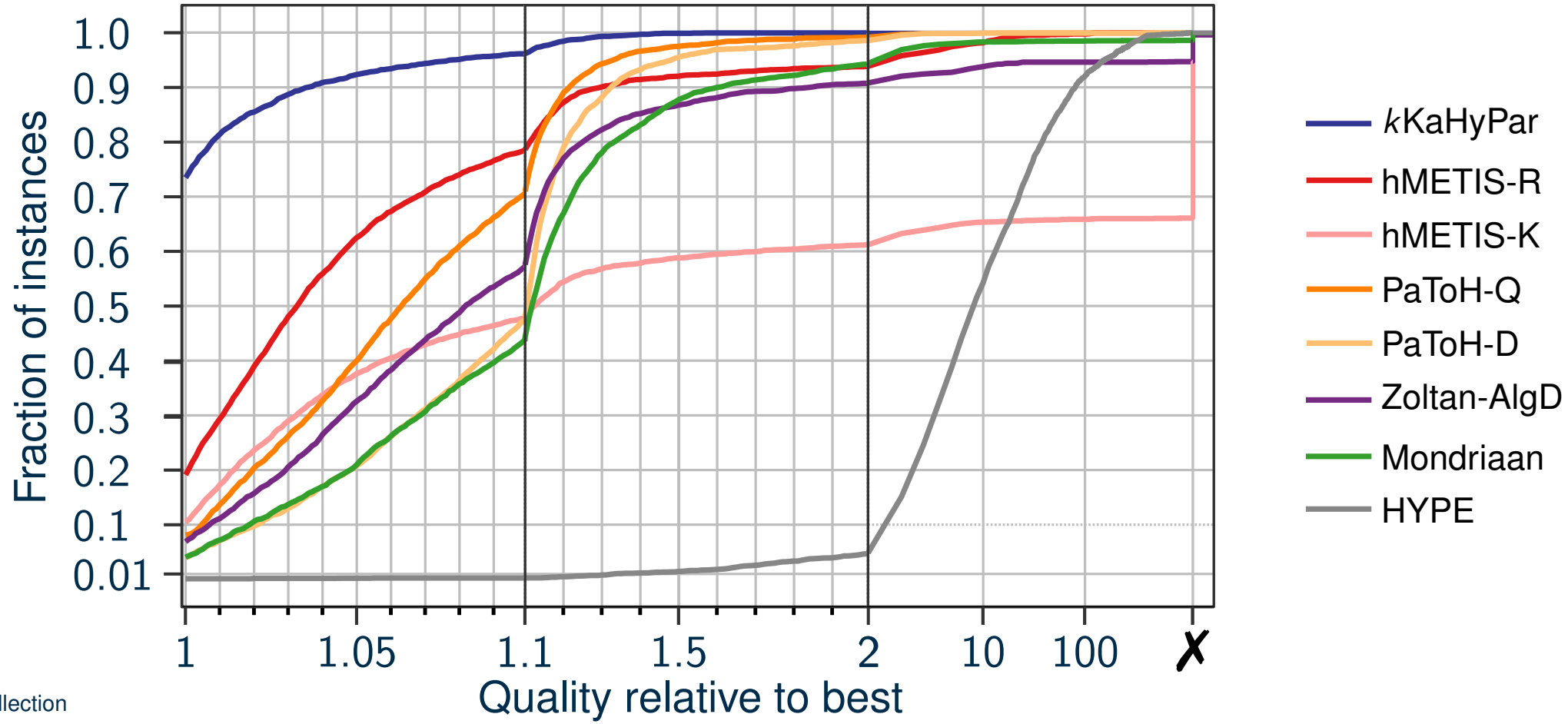


Portfolio-based Initial Partitioning
[ALENEX'16]

Connectivity Optimization: Solution Quality

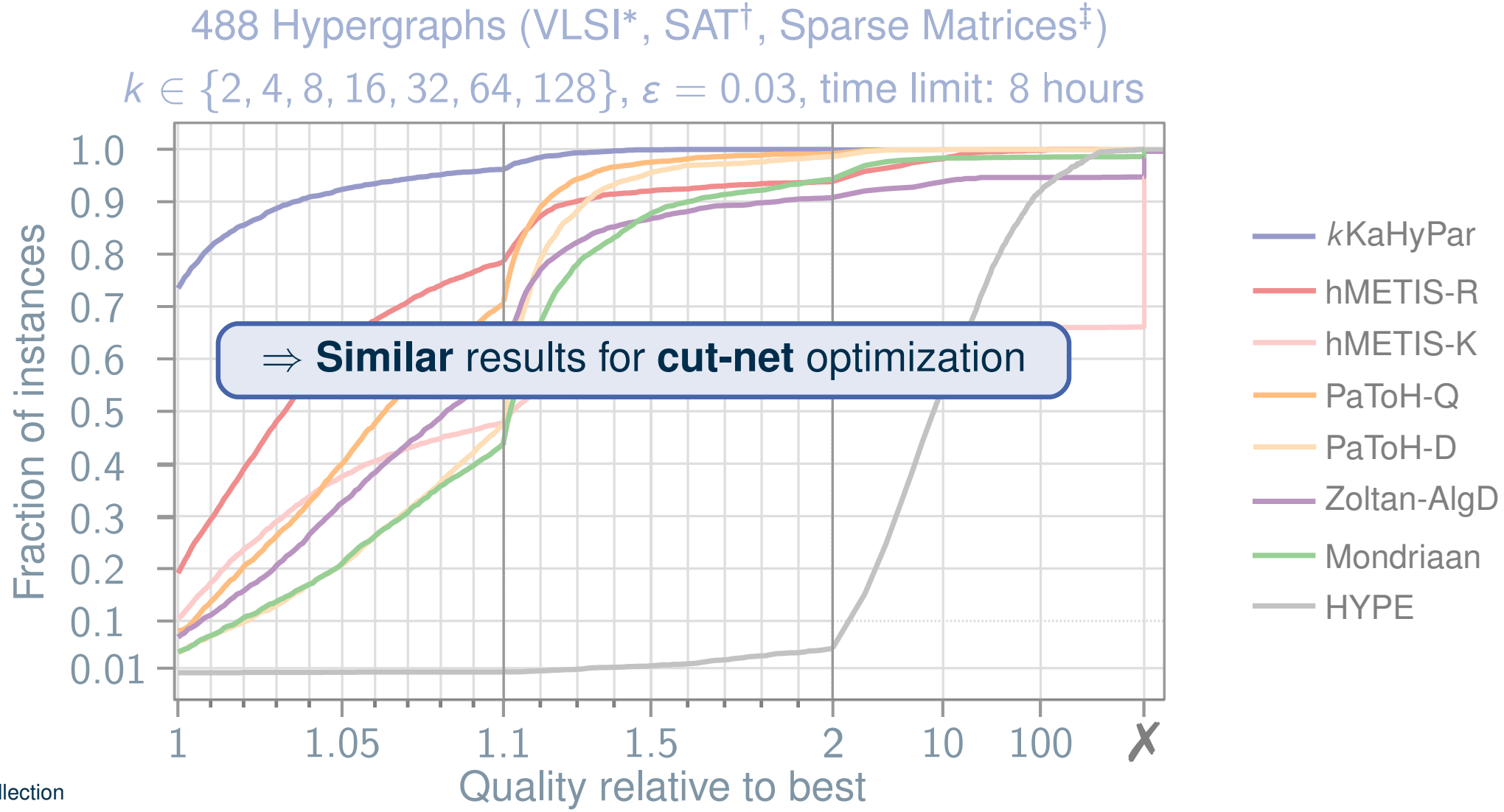
488 Hypergraphs (VLSI*, SAT†, Sparse Matrices‡)

$k \in \{2, 4, 8, 16, 32, 64, 128\}$, $\varepsilon = 0.03$, time limit: 8 hours



* ISPD 1998, DAC 2012
† SAT Competition 2014
‡ SuiteSparse Matrix Collection

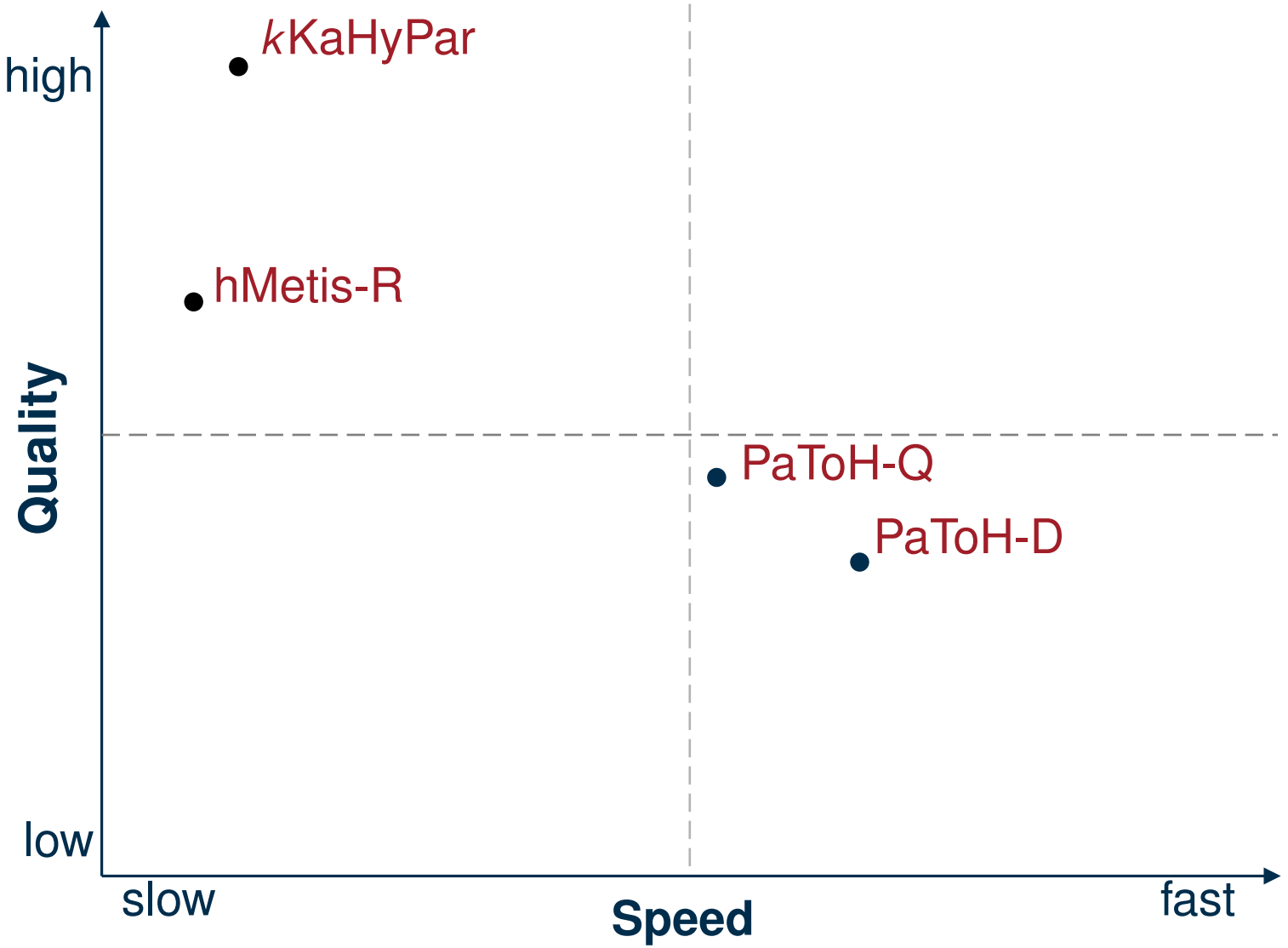
Connectivity Optimization: Solution Quality



* ISPD 1998, DAC 2012
† SAT Competition 2014
‡ SuiteSparse Matrix Collection

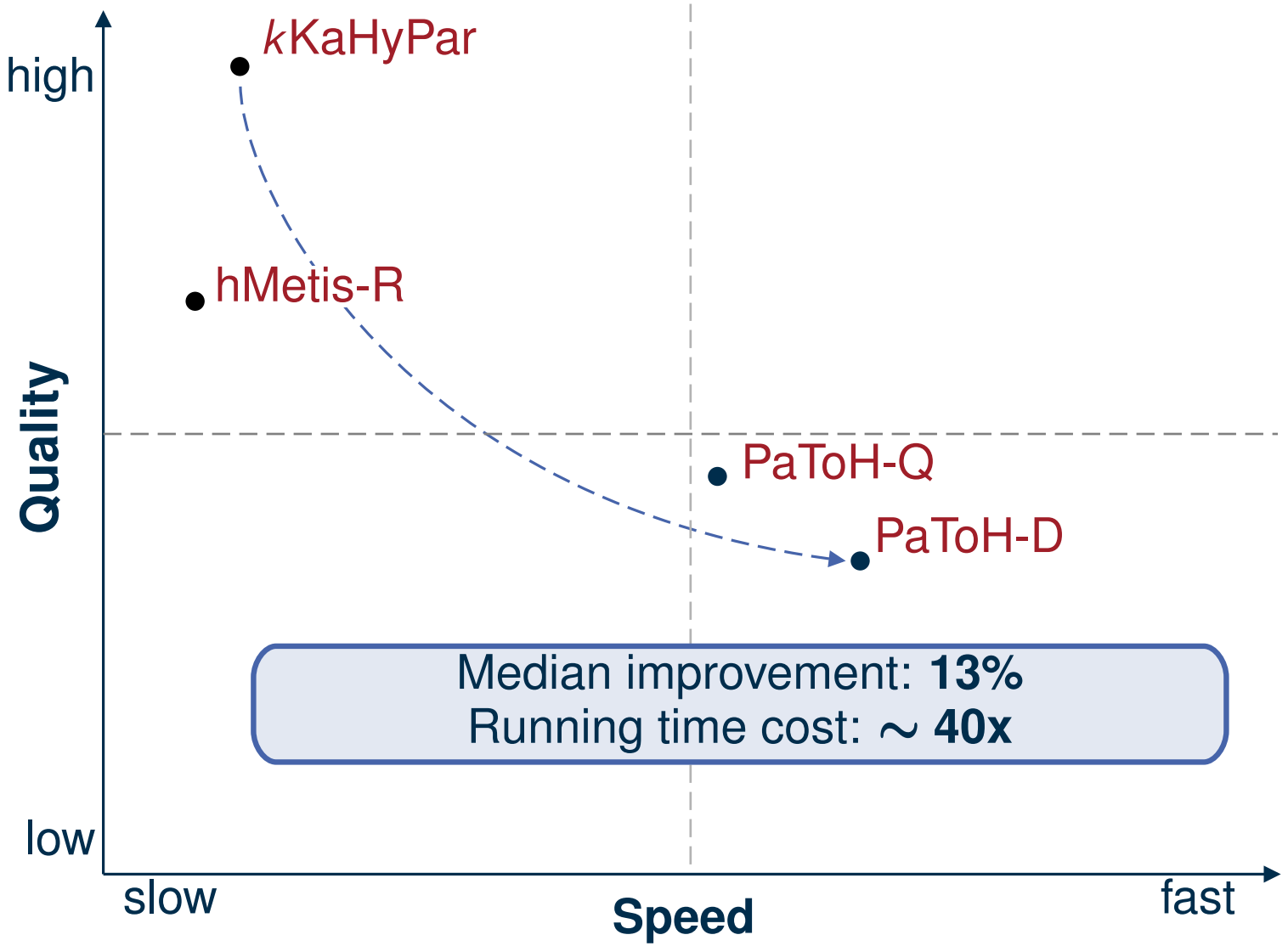
Time–Quality Trade-Off Landscape

Sequential

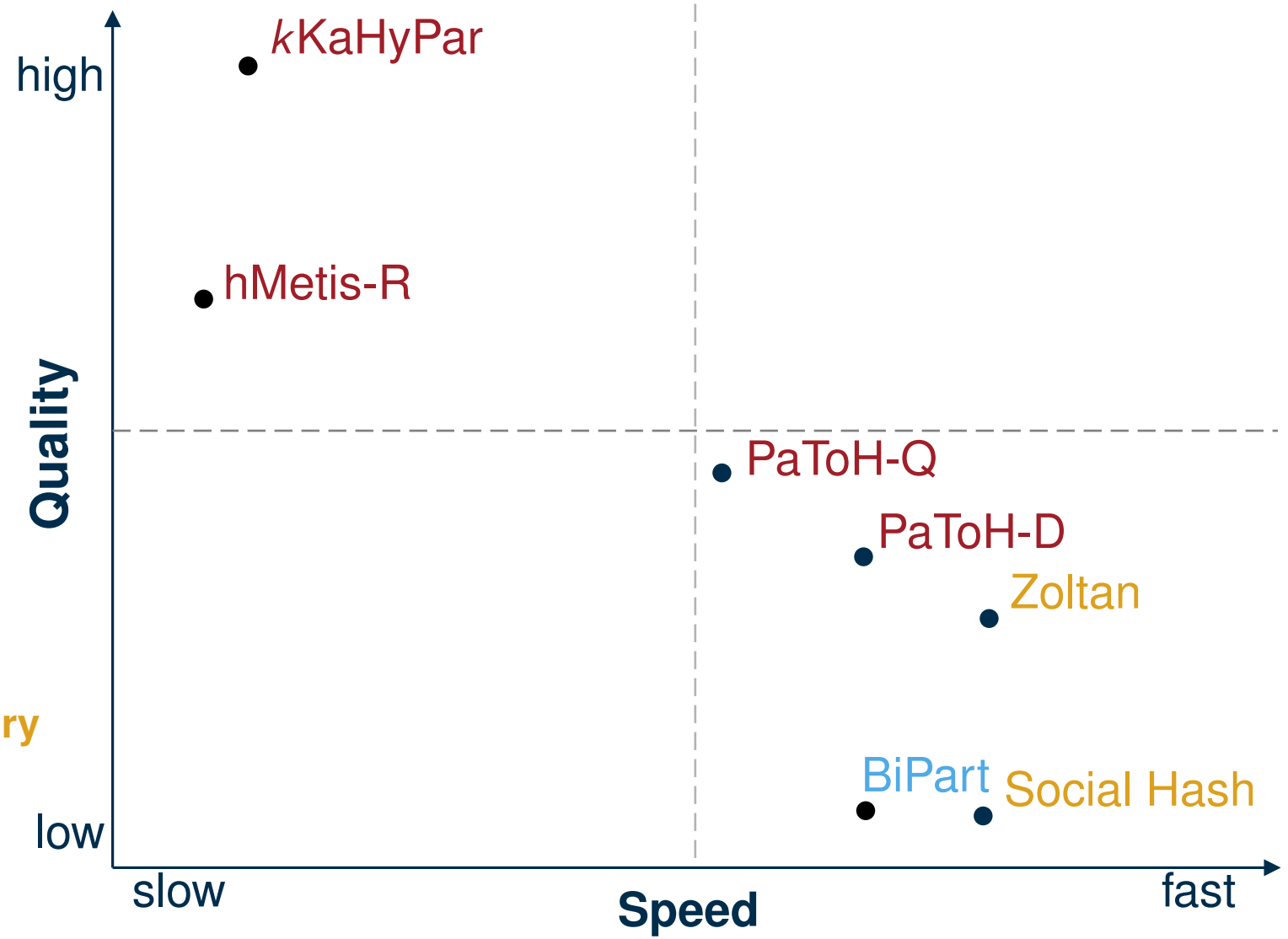


Time–Quality Trade-Off Landscape

Sequential



Time–Quality Trade-Off Landscape

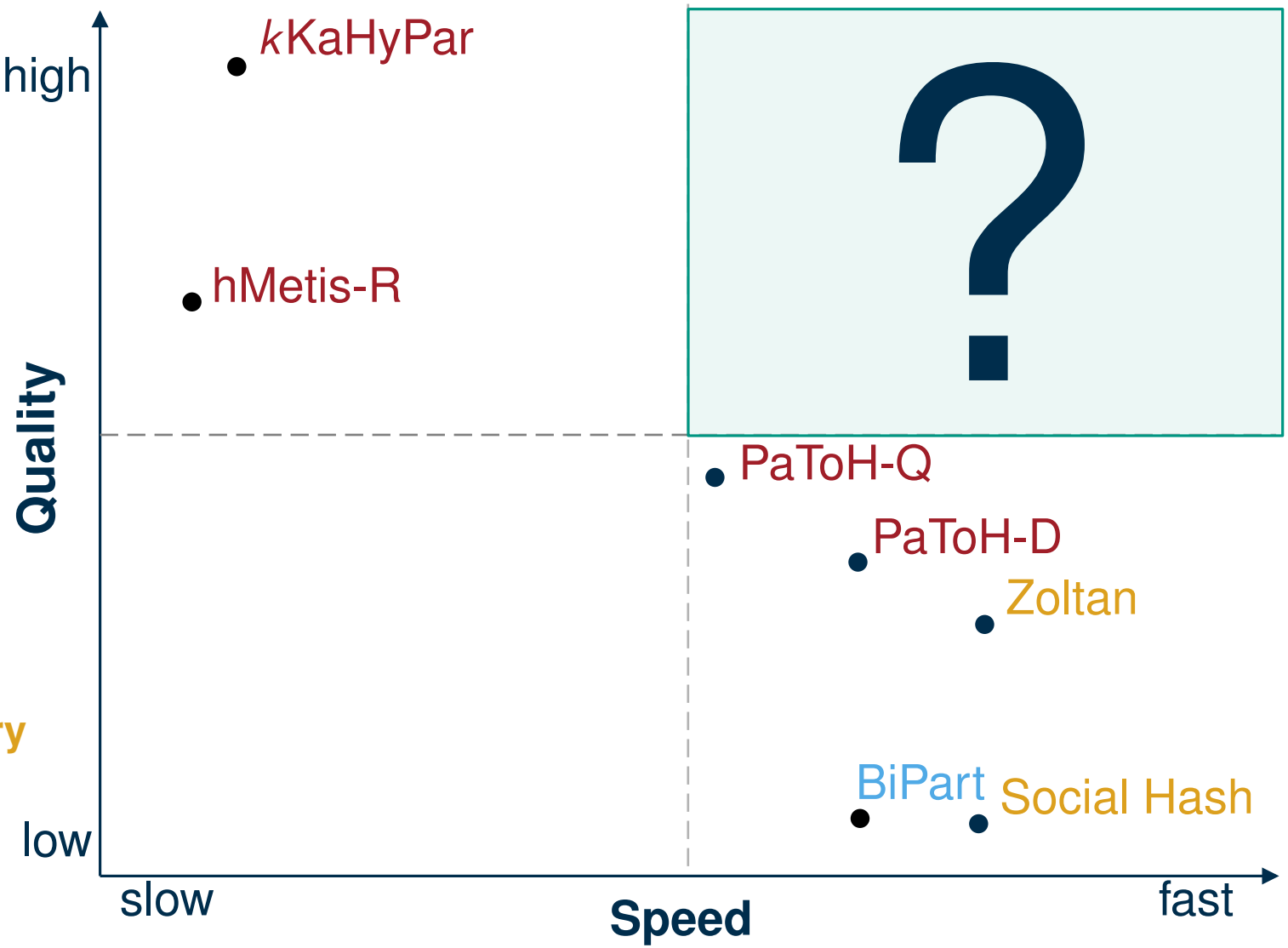


Sequential

Shared Memory

Distributed Memory

Time–Quality Trade-Off Landscape

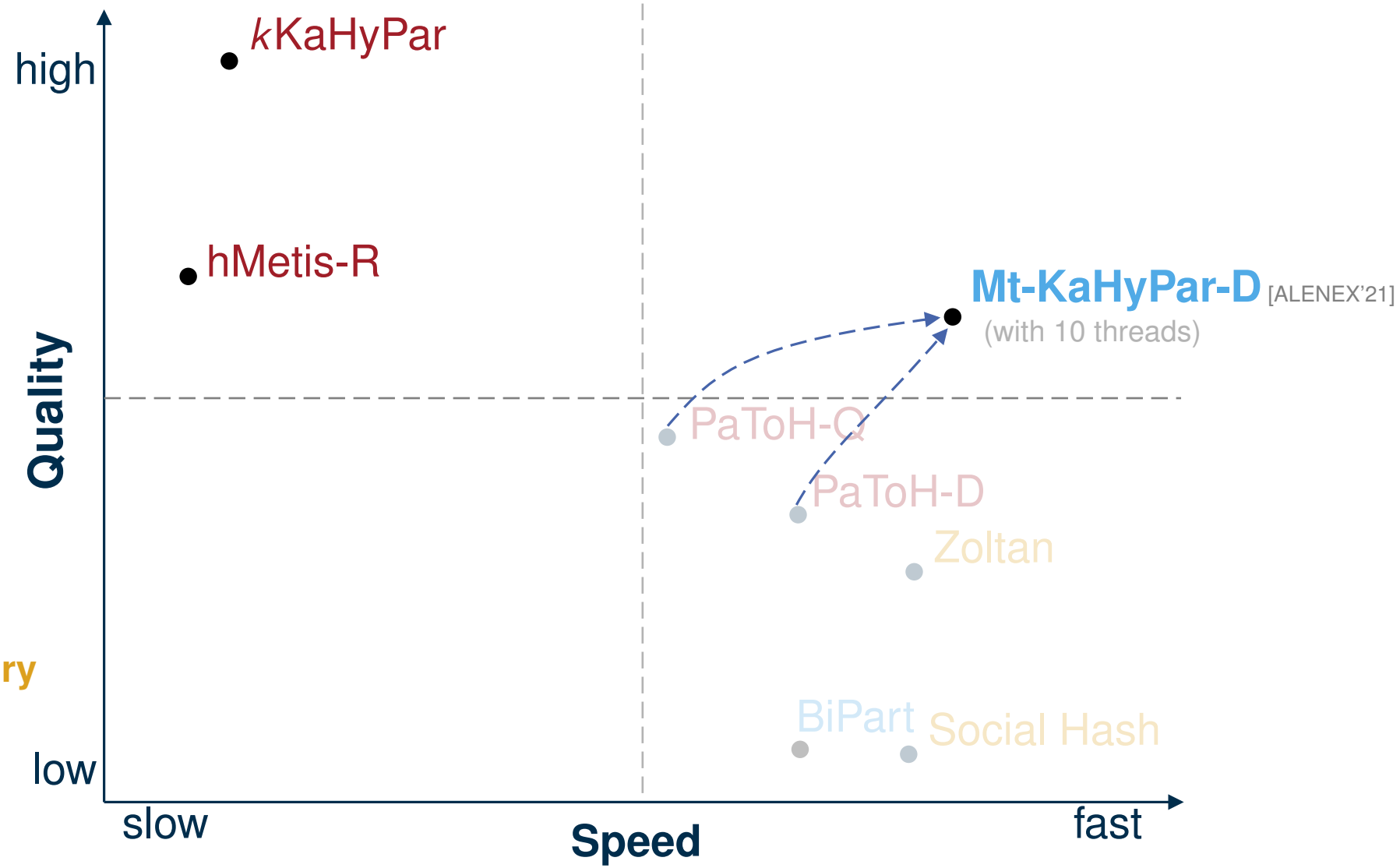


Sequential

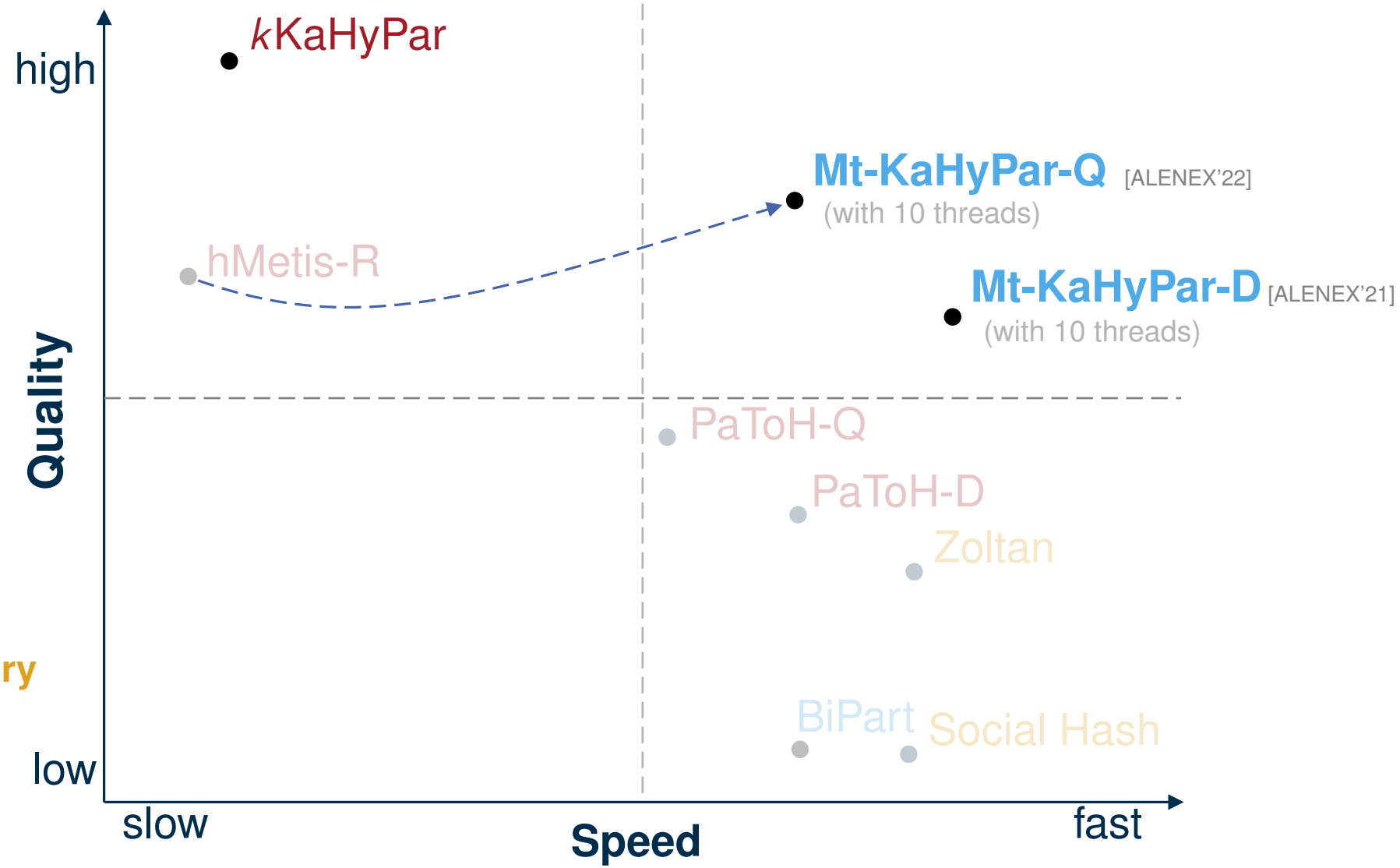
Shared Memory

Distributed Memory

Time–Quality Trade-Off Landscape



Time–Quality Trade-Off Landscape

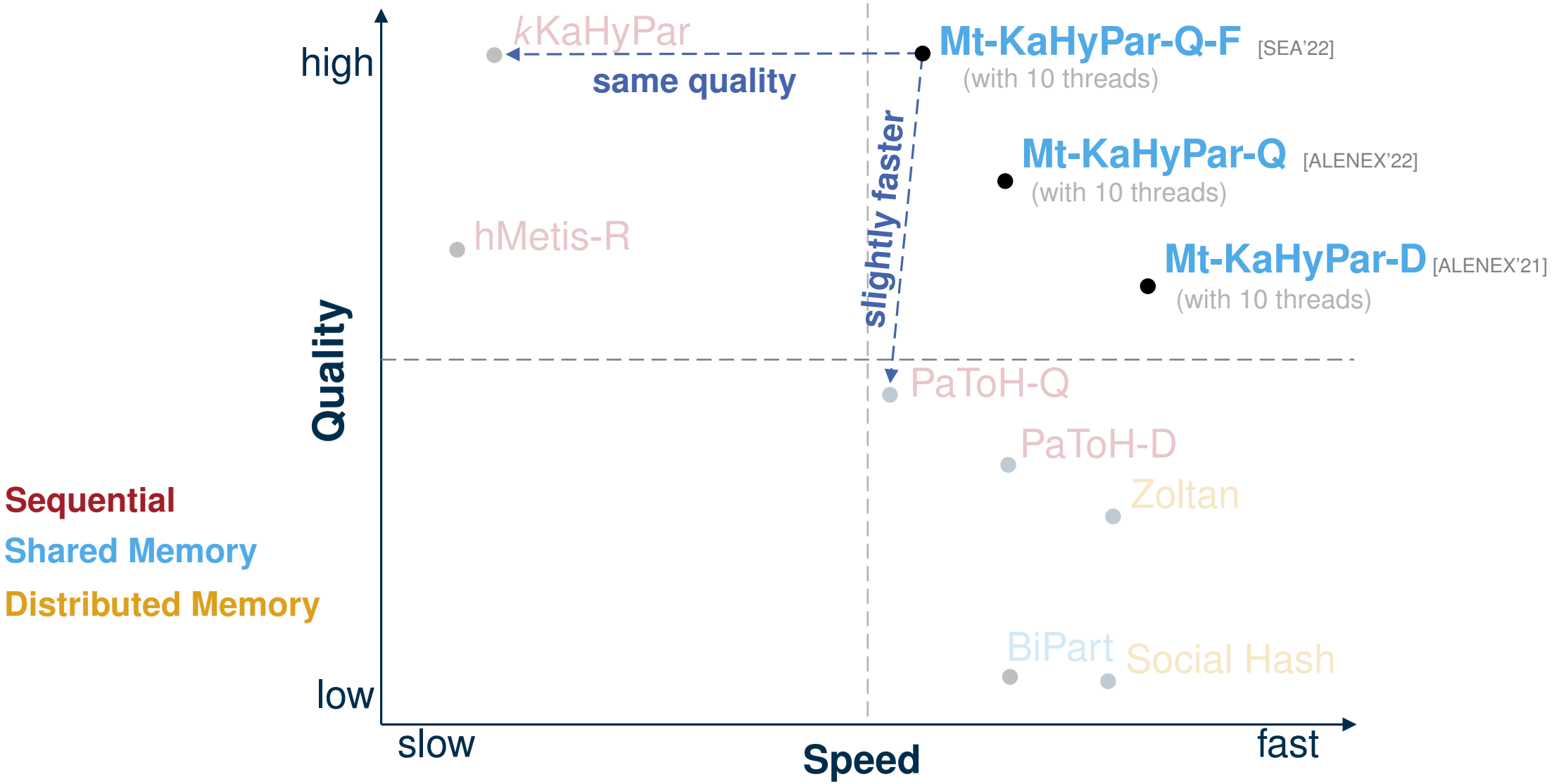


Sequential

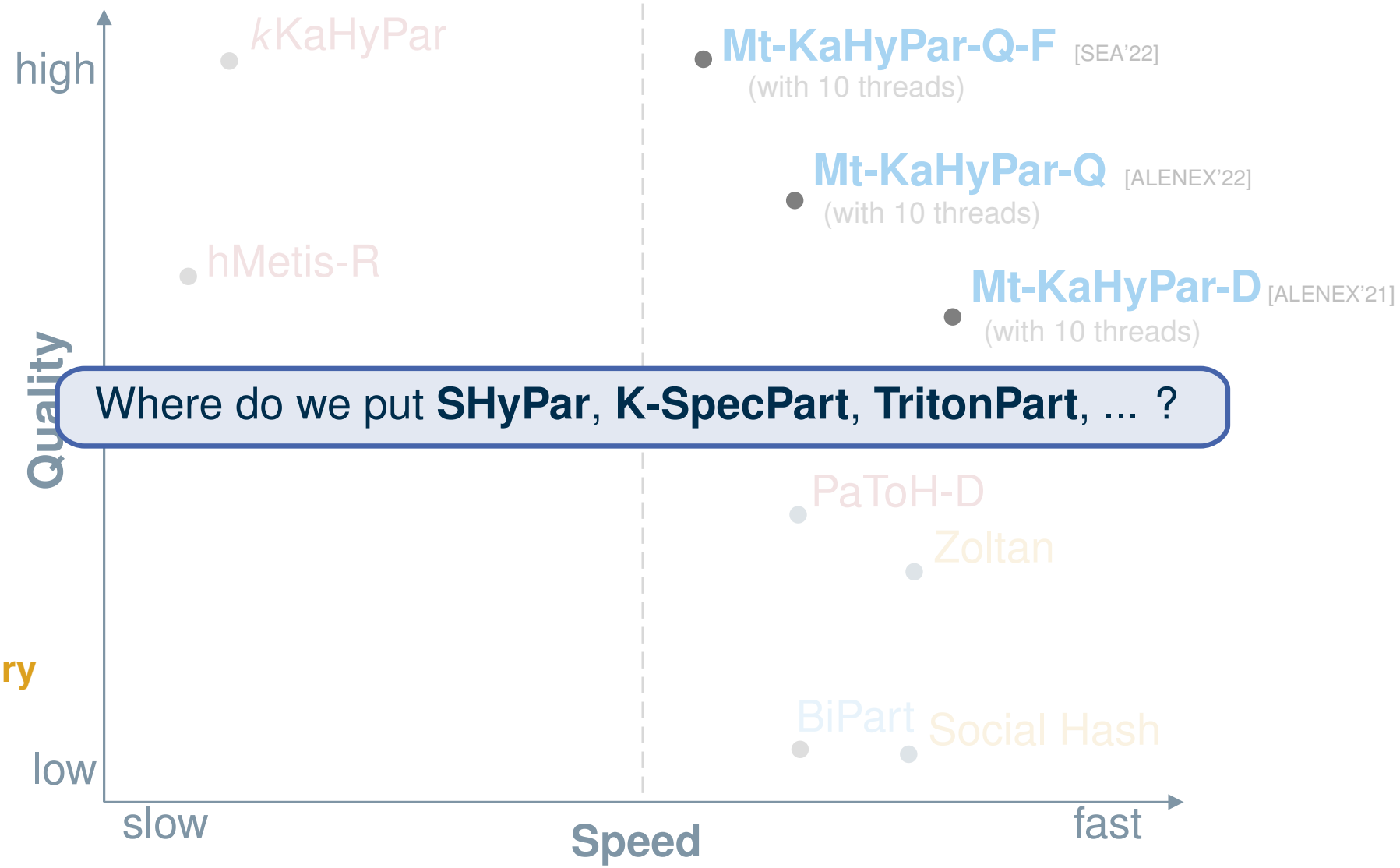
Shared Memory

Distributed Memory

Time–Quality Trade-Off Landscape



Time–Quality Trade-Off Landscape (as of 2022)



Sequential

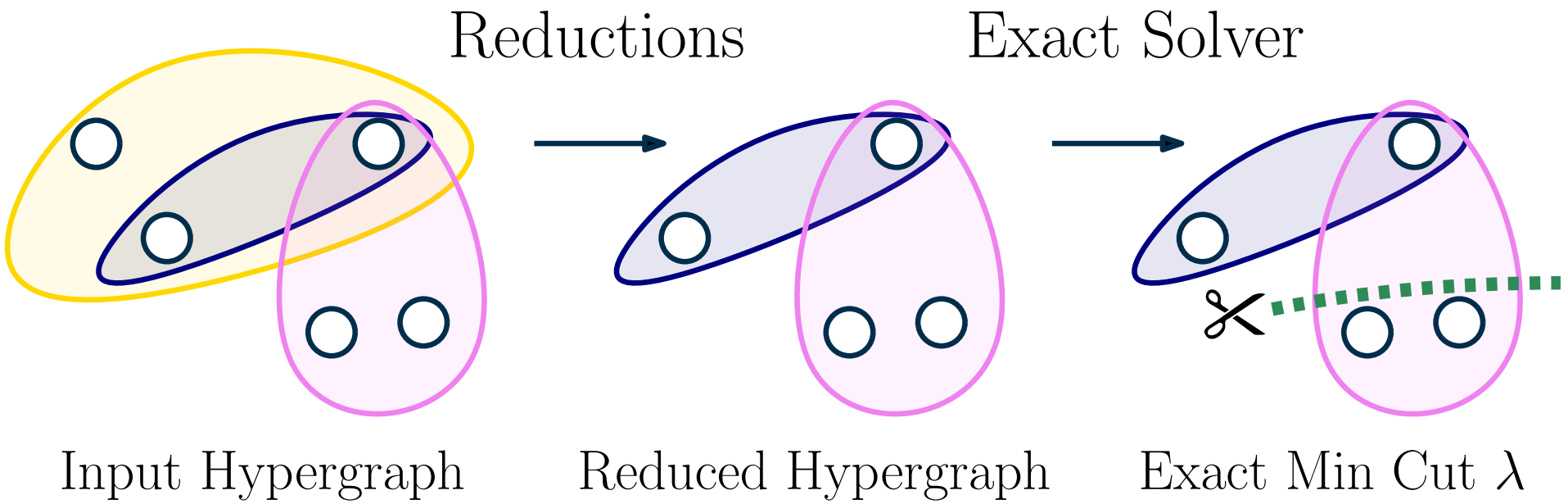
Shared Memory

Distributed Memory

Can we **scale** **exact minimum cut** computation to large hypergraphs?

HeiCut: Overall Approach [ALENEX'26]

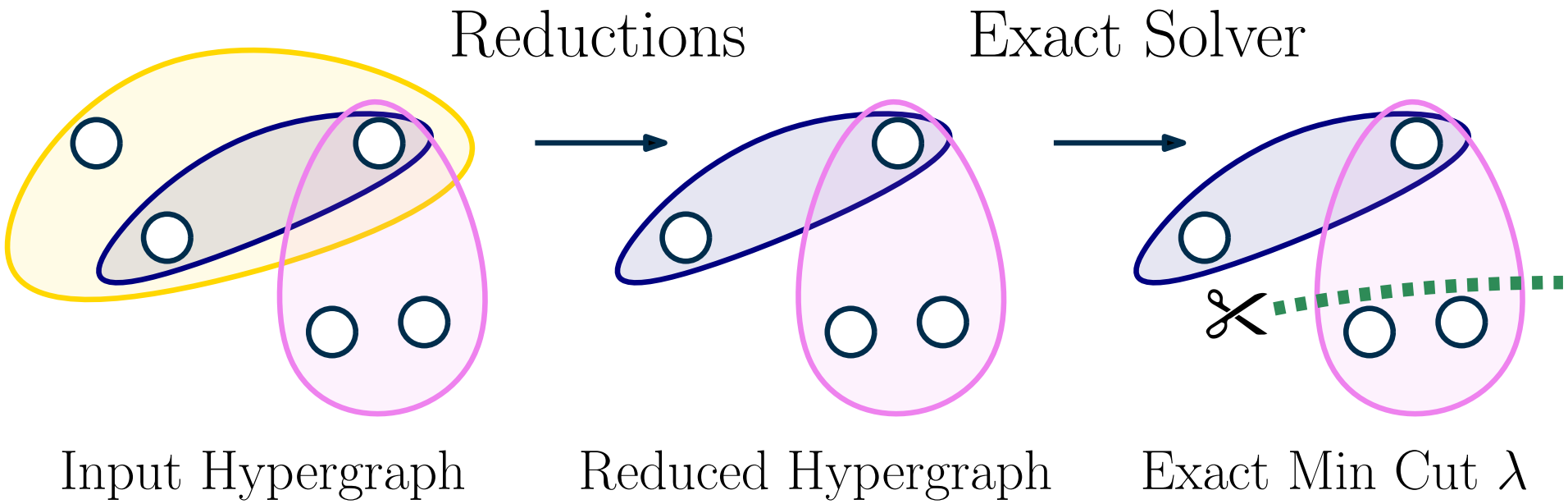
- **Key idea:** Aggressively shrink hypergraph using provably exact reduction rules
- Terminate early if the minimum cut is determined ($|V| = 1$ or $|E| = 0$), or
- Solve using an ordering-based exact solver



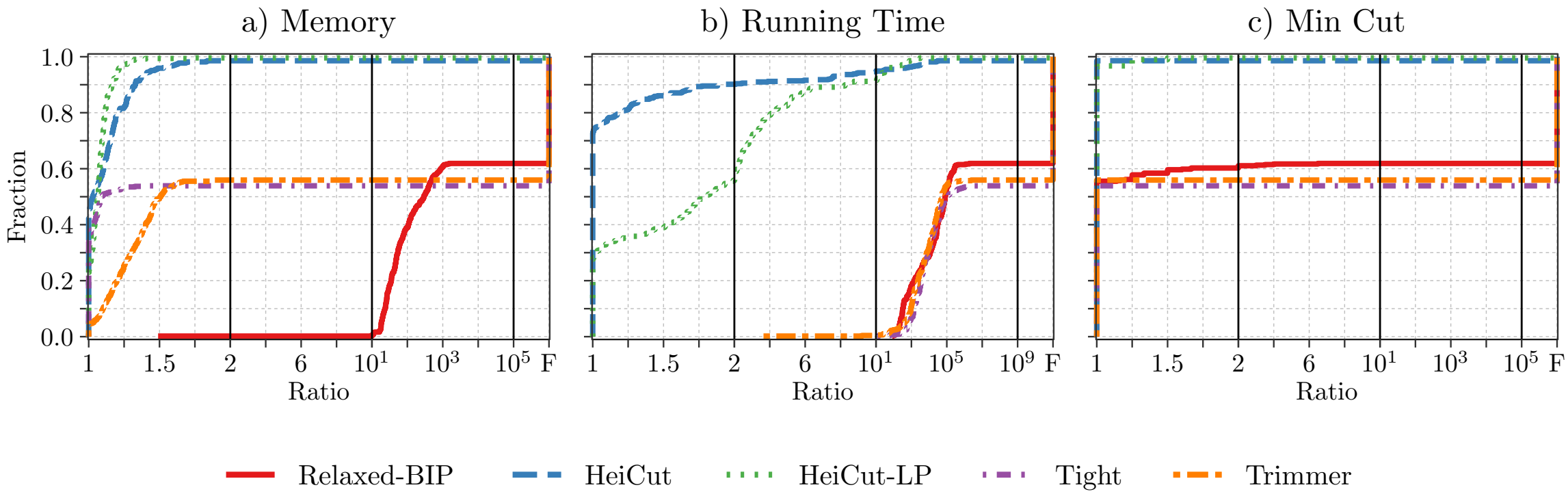
HeiCut: Overall Approach [ALENEX'26]

Preserve the minimum cut value

- **Key idea:** Aggressively shrink hypergraph using provably exact reduction rules
- Terminate early if the minimum cut is determined ($|V| = 1$ or $|E| = 0$), or
- Solve using an ordering-based exact solver



Experimental Results



C. Chekuri and C. Xu,
Computing Minimum Cuts in Hypergraphs,
SODA 2017

- HeiCut solves 98.6% of instances vs. Trimmer's 55%
- HeiCut is over 1000× faster than Trimmer on 85% of instances
- Fully reduced (single vertex) in 85% of instances → min cut without invoking exact solver

Open-Source Frameworks

KaHyPar – Sequential HGP (GPL v3)

- Objectives: cut, connectivity
- Supports: Fixed vertices, custom block weights, memetic partitioning
- Interfaces: C, Python, Java, Julia
- `http://www.kahypar.org`

Mt-KaHyPar – Shared-Memory HGP (MIT)

- Objectives: cut, connectivity, sum-of-external-degrees, Steiner tree
- Supports: Fixed vertices, custom block weights, deterministic & large-k partitioning
- Interfaces: C, Python, Julia
- `https://github.com/kahypar/mt-kahypar`

HeiCut – Exact minimum cuts at scale (MIT)

- `https://github.com/HeiCut/HeiCut`