Evolutionary Hypergraph Partitioning

Presentation · December 11, 2017

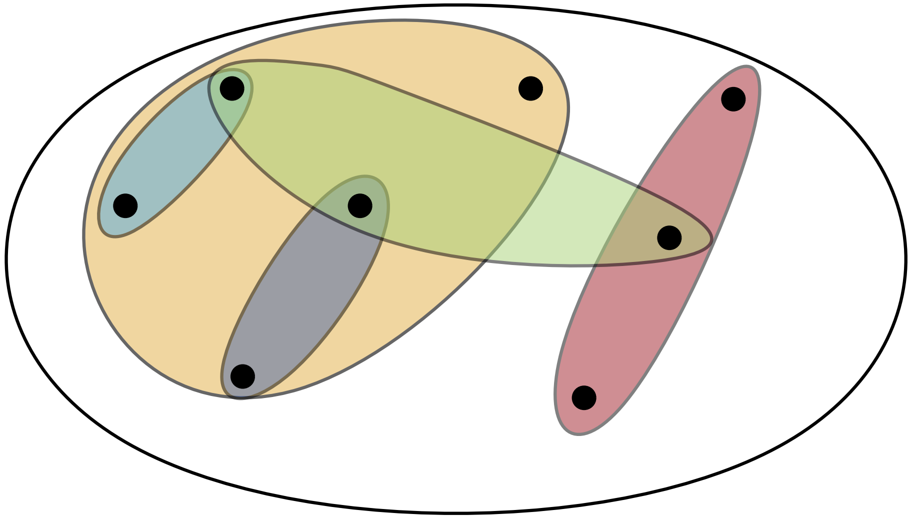
Robin Andre

INSTITUTE OF THEORETICAL INFORMATICS ·

Problem

Hypergraph Partitioning

Given a hypergraph $H := (V, E, c, w)$ a partition is a distribution of the nodes in k disjoint blocks $V_1, ..., V_k$. A partition is balanced if
$\forall\ 1 \leq i \leq k\ :\ c(V_i) \leq (1+\epsilon)\lceil \frac{c(V)}{k} \rceil$ for an imbalance parameter ϵ.

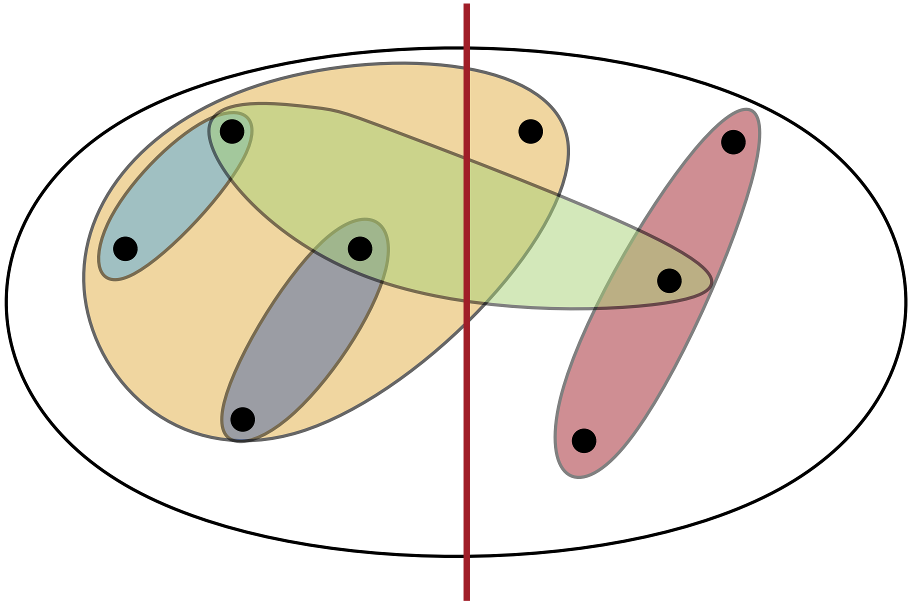


Problem

Hypergraph Partitioning

Given a hypergraph $H := (V, E, c, w)$ a partition is a distribution of the nodes in k disjoint blocks $V_1, ..., V_k$. A partition is balanced if $\forall\ 1 \leq i \leq k\ :\ c(V_i) \leq (1 + \epsilon)\lceil \frac{c(V)}{k} \rceil$ for an imbalance parameter ϵ.

$k = 2$; $cut = 2$; $(\lambda - 1) = 2$

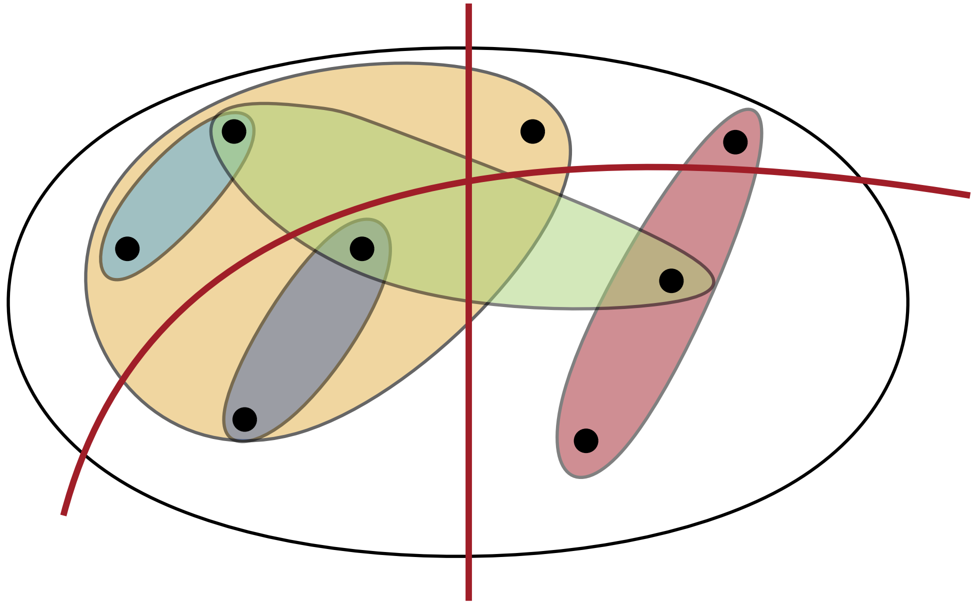


Problem

Hypergraph Partitioning

Given a hypergraph $H := (V, E, c, w)$ a partition is a distribution of the nodes in k disjoint blocks $V_1, ..., V_k$. A partition is balanced if $\forall\ 1 \leq i \leq k\ :\ c(V_i) \leq (1 + \epsilon)\lceil \frac{c(V)}{k} \rceil$ for an imbalance parameter ϵ.

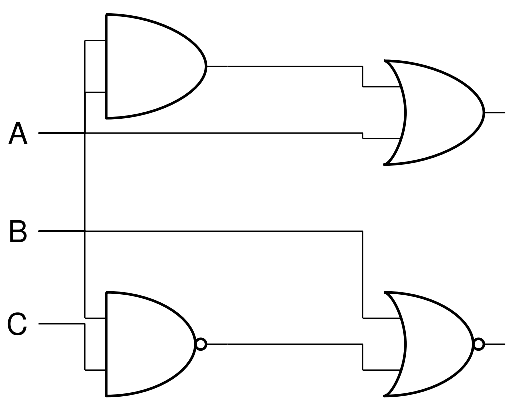
$k = 4; cut = 3; (\lambda - 1) = 5$



Motivation

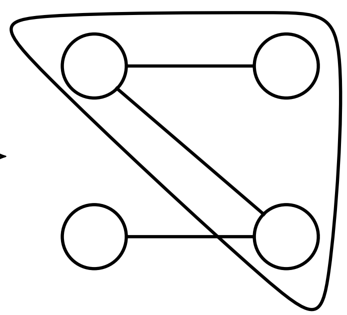
- Hypergraph partitioning is NP-hard
- Evolutionary Algorithms are generating high quality solutions
- Many applications benefit from the best possible solution

VLSI Design

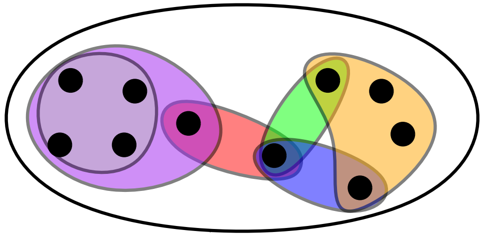


Scientific Computing

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \Rightarrow$$



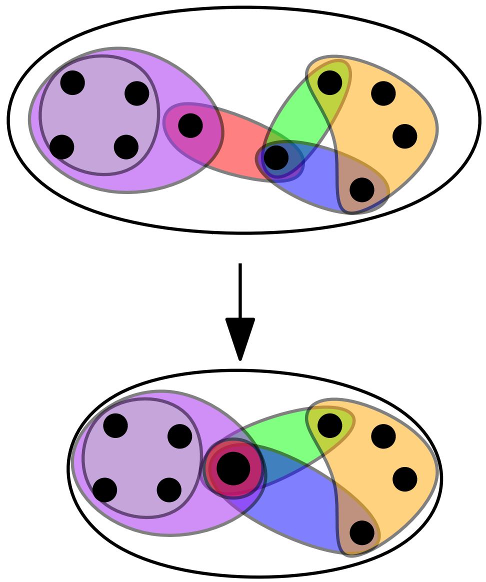
KaHyPar - Multilevel Partitioner



Coarsening:

- H is reduced to a smaller problem H_C
- only one node is contracted per step
- until H_C is sufficiently small

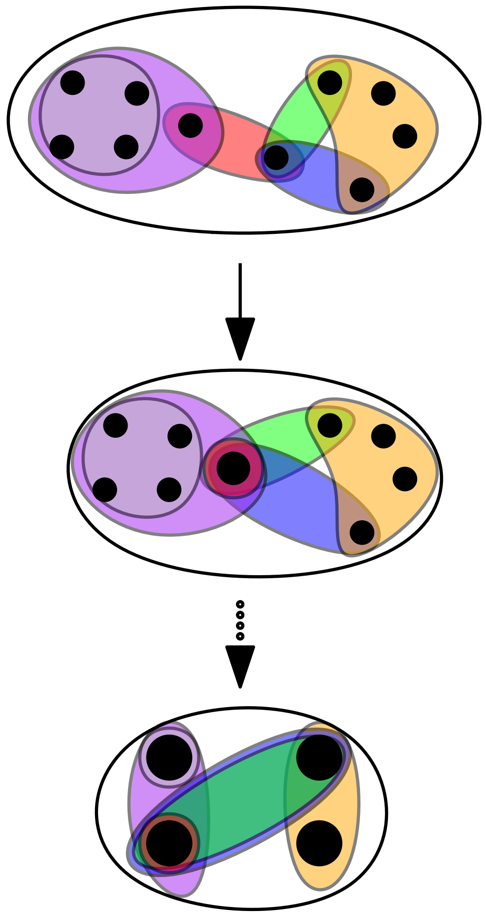
KaHyPar - Multilevel Partitioner



Coarsening:

- H is reduced to a smaller problem H_C
- only one node is contracted per step
- until H_C is sufficiently small

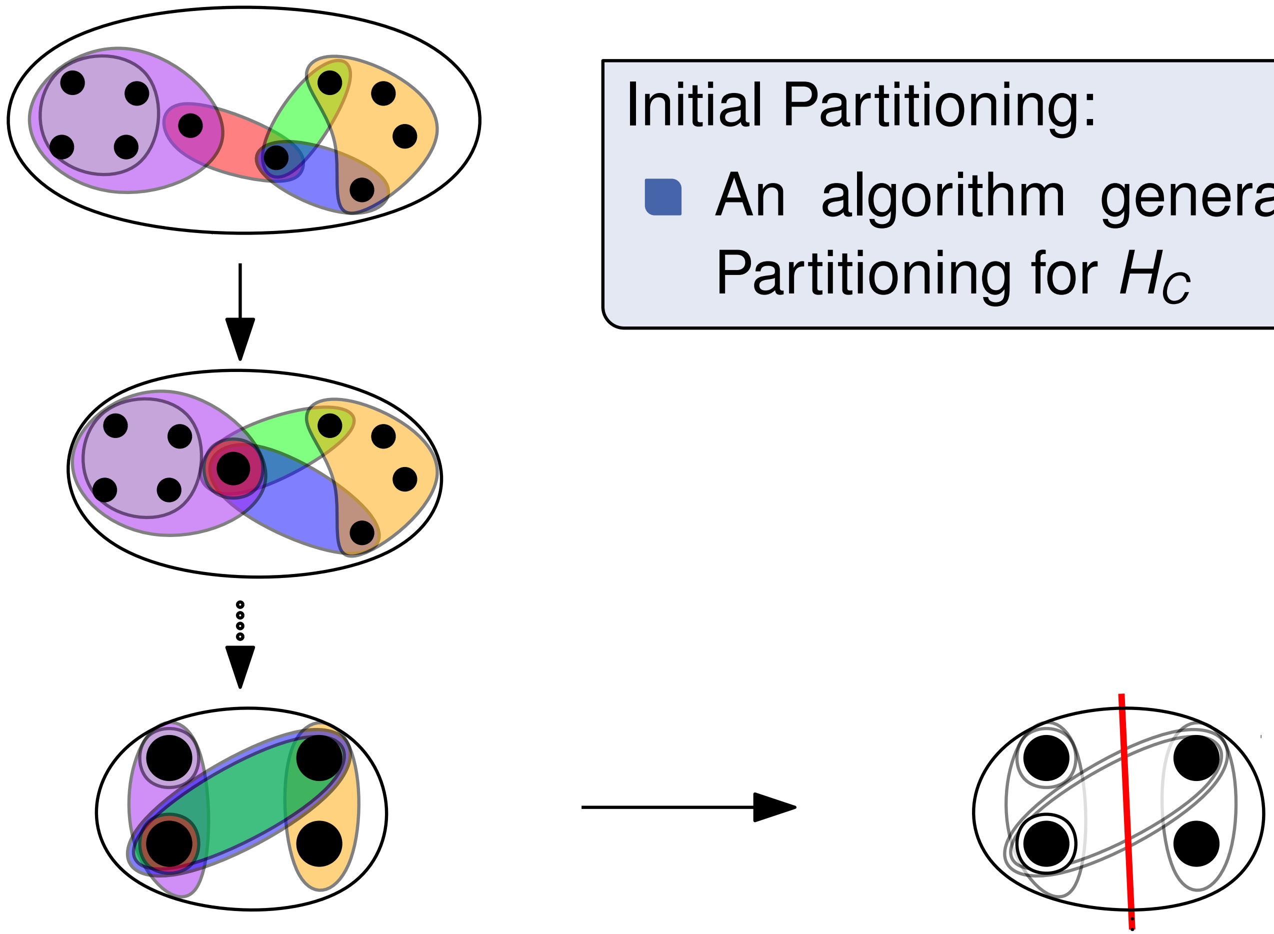
KaHyPar - Multilevel Partitioner



Coarsening:

- H is reduced to a smaller problem H_C
- only one node is contracted per step
- until H_C is sufficiently small

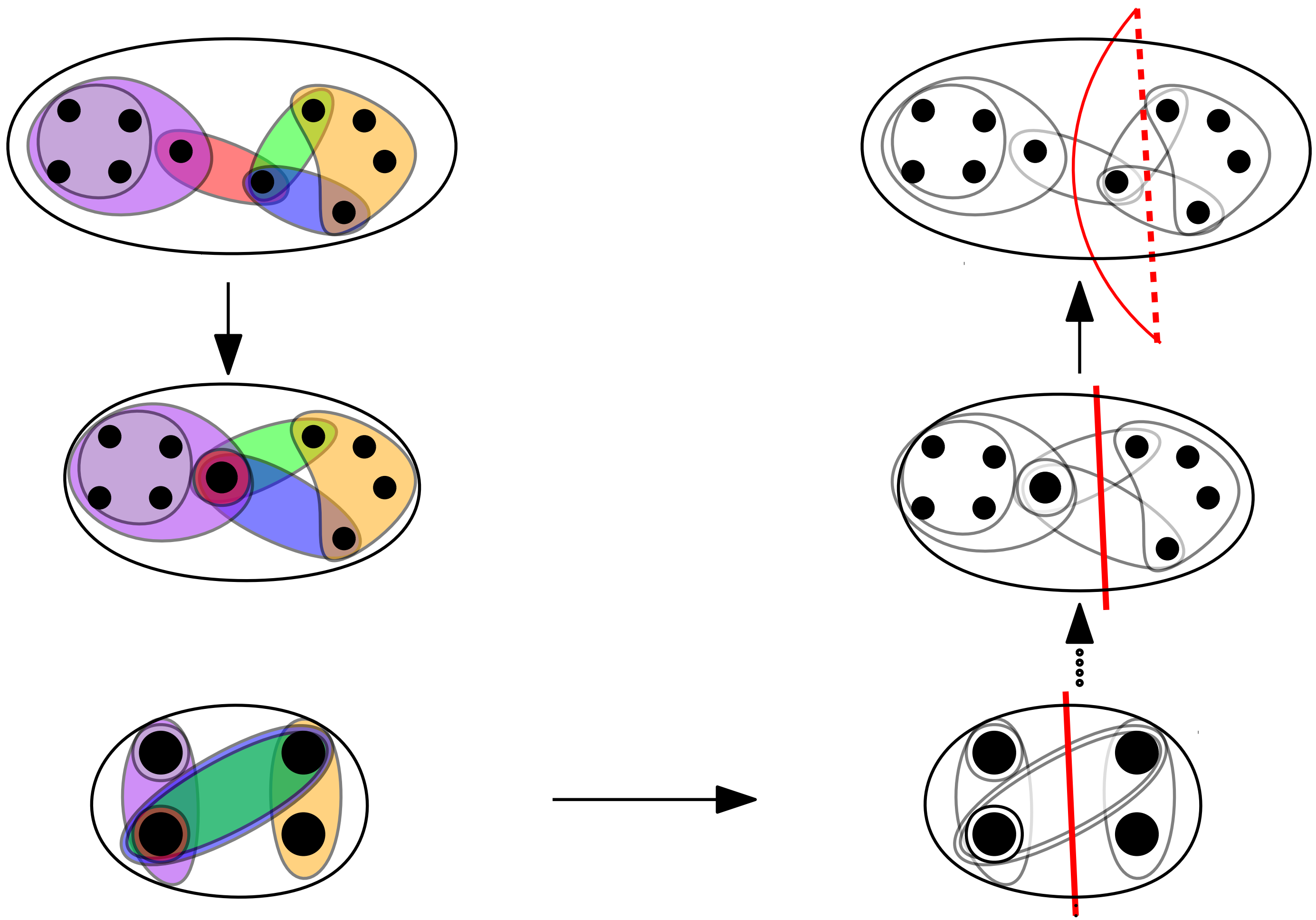
KaHyPar - Multilevel Partitioner



Initial Partitioning:

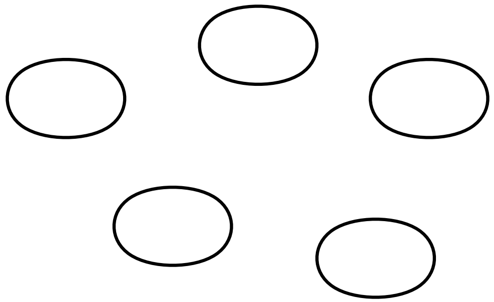
- An algorithm generates an Initial Partitioning for H_C

KaHyPar - Multilevel Partitioner



Evolutionary Process

- Choose individuals for recombination



Evolutionary Process

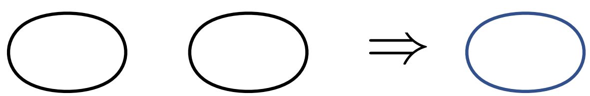
- Choose individuals for recombination

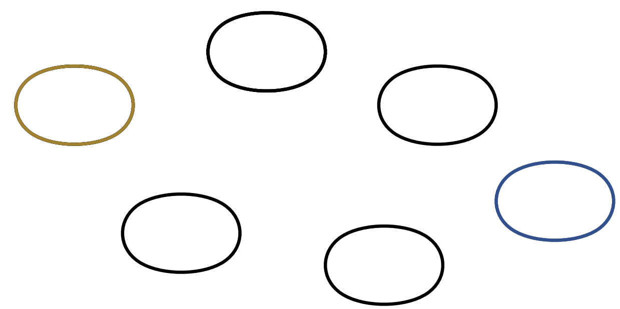
Evolutionary Process

- Choose individuals for recombination
- Generate offspring O

Evolutionary Process

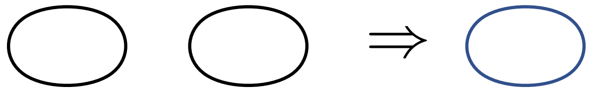
- Choose individuals for recombination
- Generate offspring O
- Perform mutations M

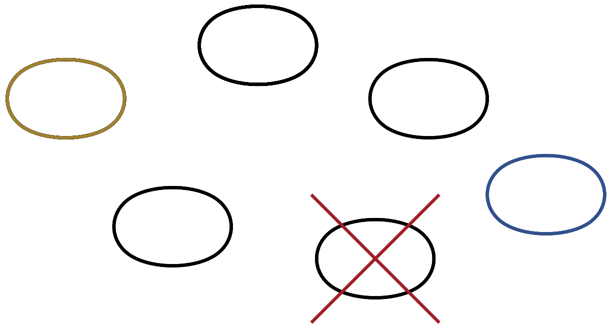




Evolutionary Process

- Choose individuals for recombination
- Generate offspring O
- Perform mutations M
- Select survivors



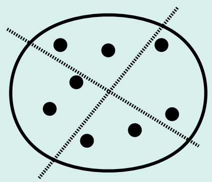


Evolutionary Algorithm

KaHyPar generates multiple partitions

dynamic allocation δ = 15%

Evolutionary Algorithm



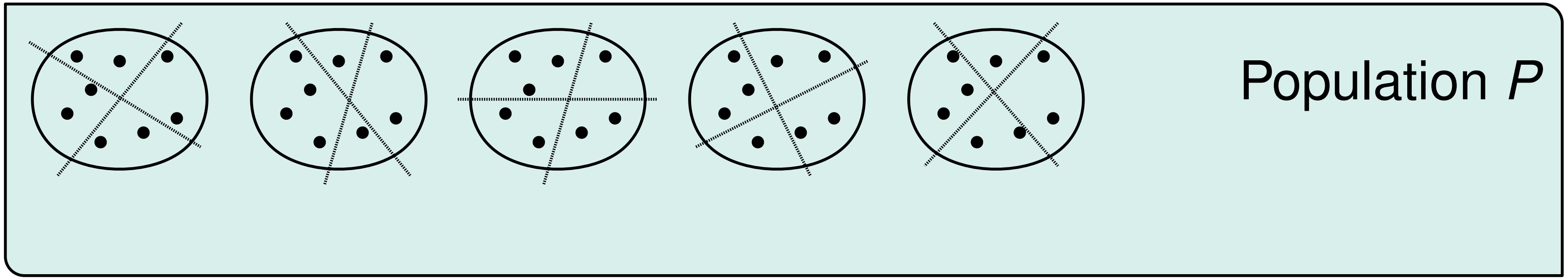
Population P

3.1s $time = 100s$ $\longrightarrow$ $\sim$ 5 iterations

KaHyPar generates multiple partitions

dynamic allocation δ = 15%

Evolutionary Algorithm

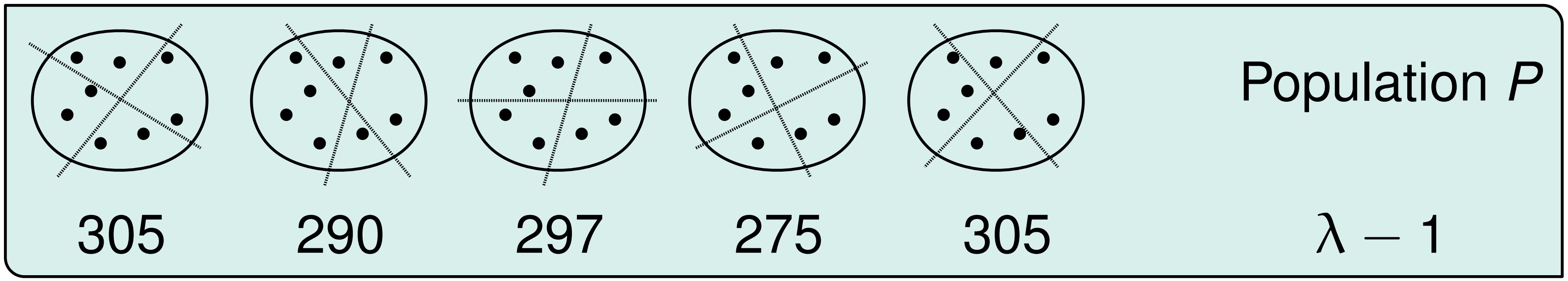


3.1s $time = 100s$ $\longrightarrow$ $\sim$ 5 iterations

KaHyPar generates multiple partitions

dynamic allocation δ = 15%

Evolutionary Algorithm

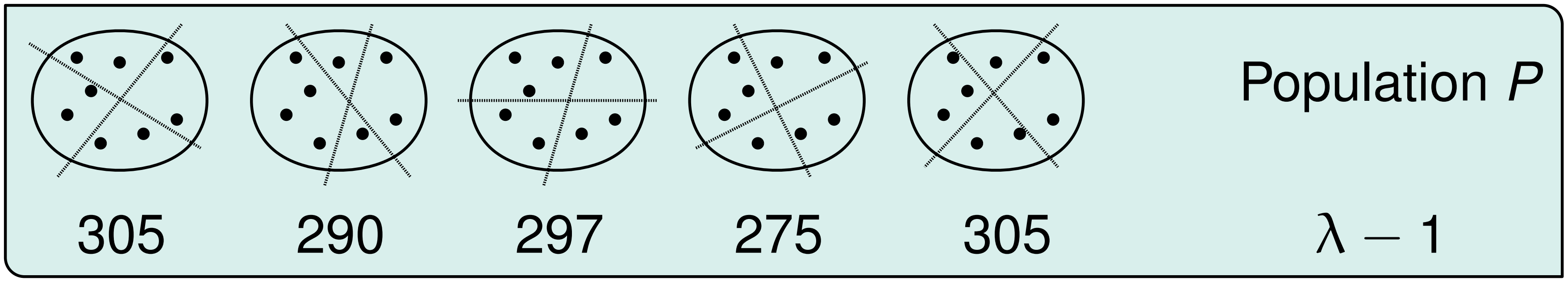


3.1s $time = 100s$ ⟶ $\sim$ 5 iterations

KaHyPar generates multiple partitions

dynamic allocation δ = 15%

Evolutionary Algorithm



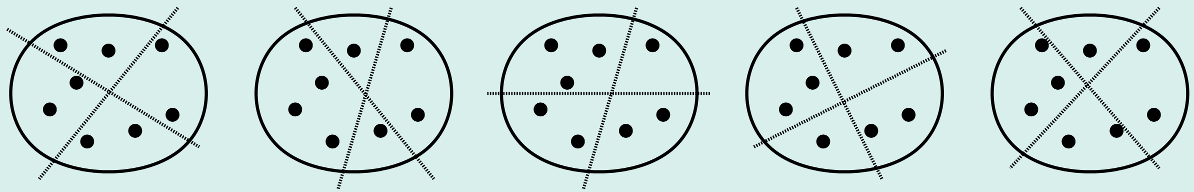
3.1s $\quad$ $time = 100s$ $\longrightarrow$ $\sim$ 5 iterations

KaHyPar generates multiple partitions

dynamic allocation $\delta = 15\%$

balances time/hypergraph size

Evolutionary Algorithm



Population P

305 290 297 275 305 $\lambda - 1$

3.1s $time = 100s$ $\longrightarrow$ $\sim$ 5 iterations

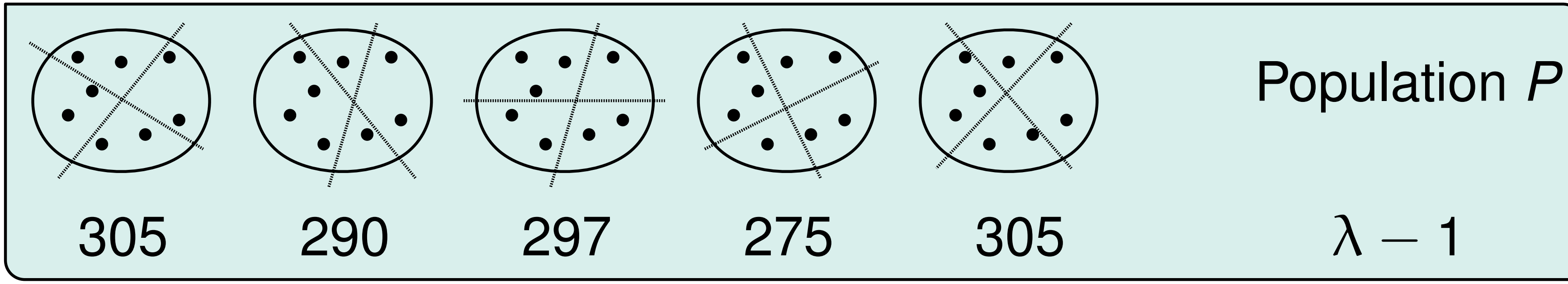
KaHyPar generates multiple partitions

dynamic allocation $\delta = 15\%$

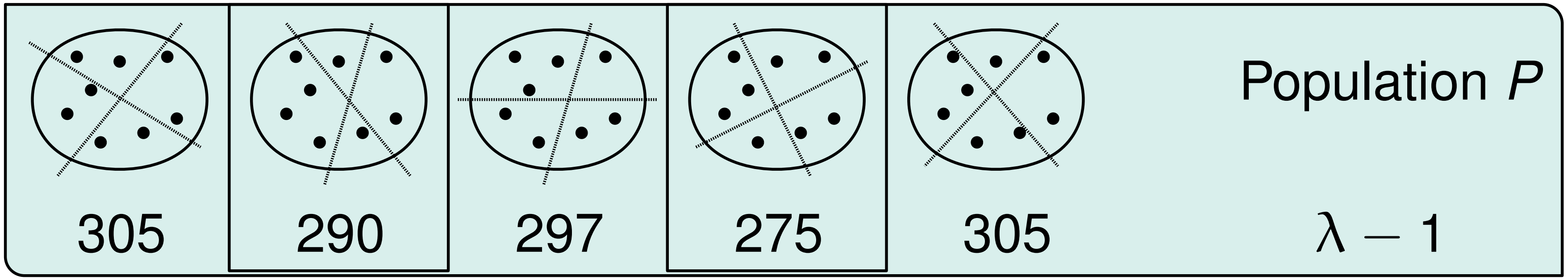
balances time/hypergraph size

high quality solutions

Tournament Selection

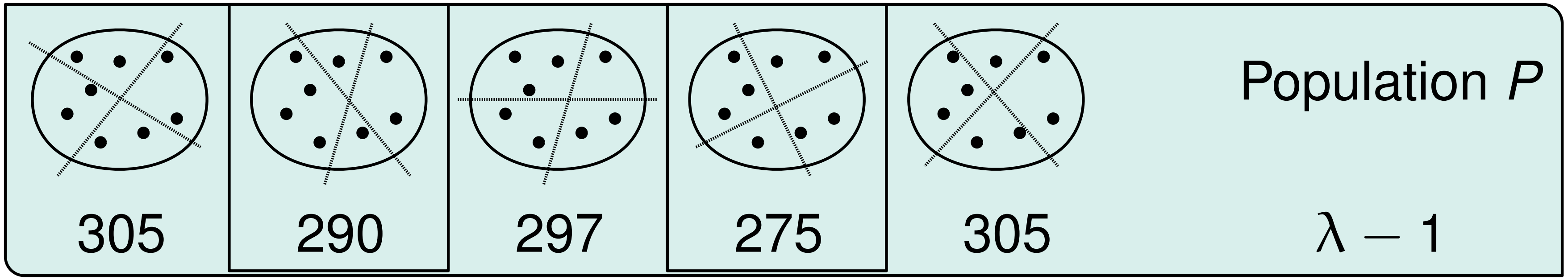


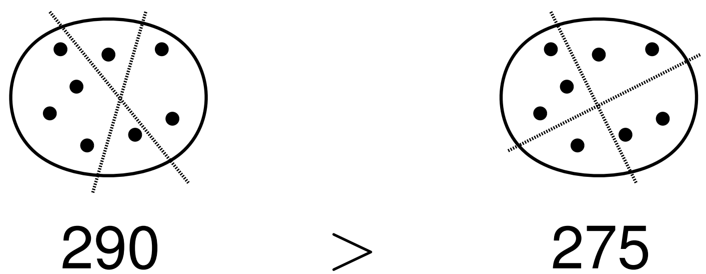
Tournament Selection



select 2 random Individuals

Tournament Selection

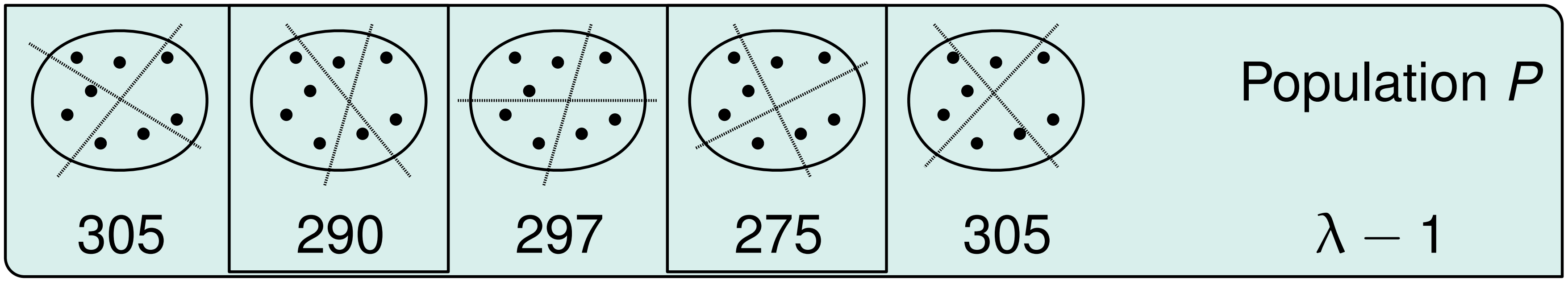


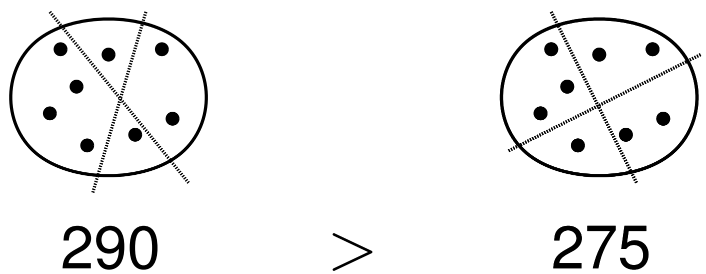


select 2 random Individuals

compare their fitness

Tournament Selection



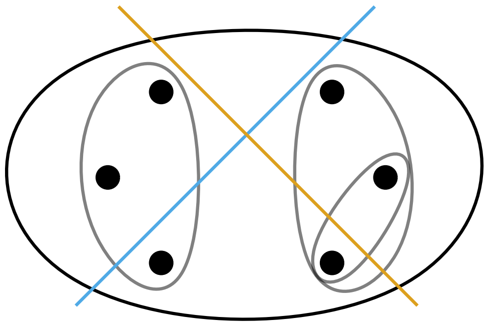


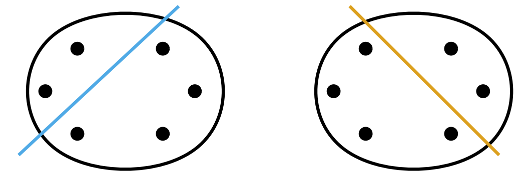
select 2 random Individuals

compare their fitness

choose the better Individual

Combine Operator



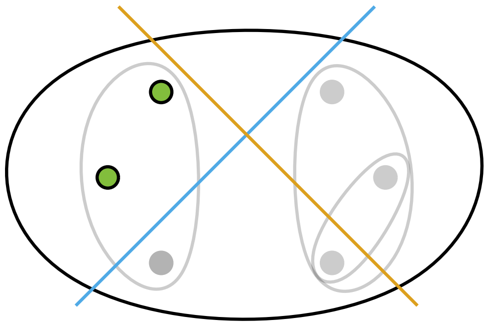


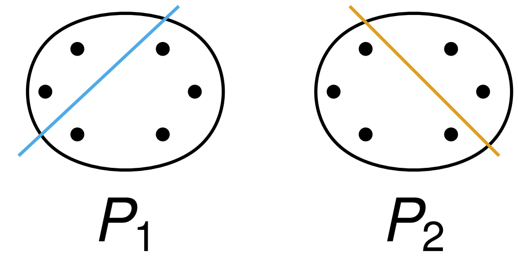
P_1 P_2

Coarsening:

- contractions must respect P_1 & P_2
- does not change solution quality

Combine Operator



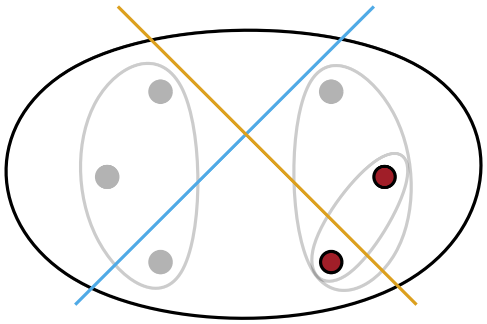


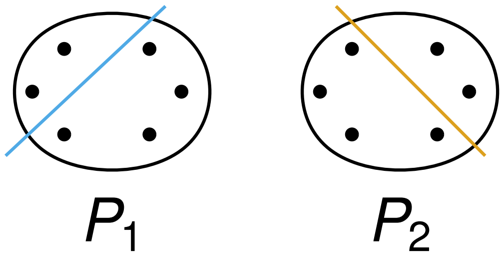
Valid Contraction

Coarsening:

- contractions must respect P_1 & P_2
- does not change solution quality

Combine Operator



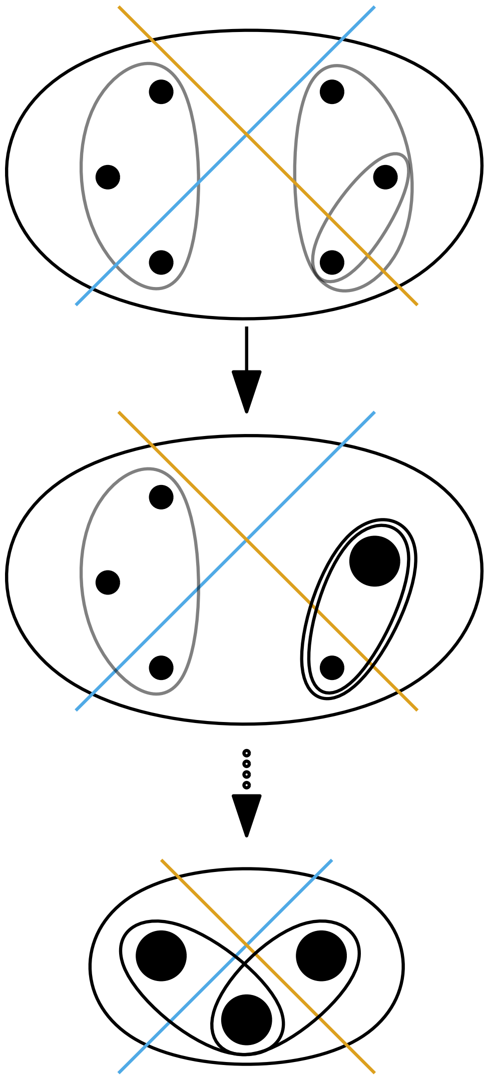


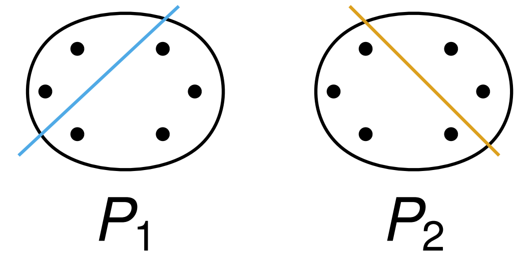
Invalid Contraction

Coarsening:

- contractions must respect P_1 & P_2
- does not change solution quality

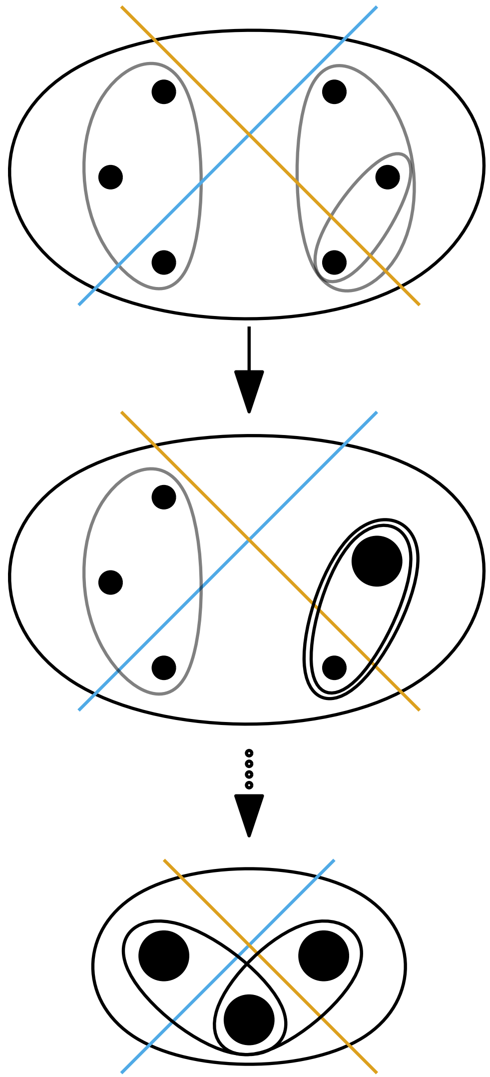
Combine Operator

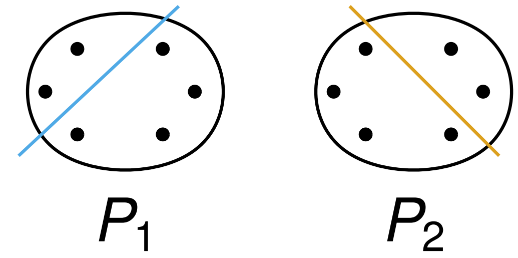




Coarsening:
- contractions must respect P_1 & P_2
- does not change solution quality

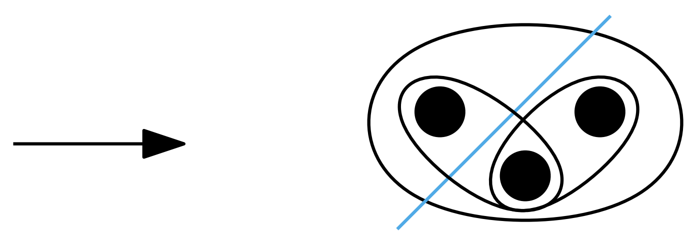
Combine Operator



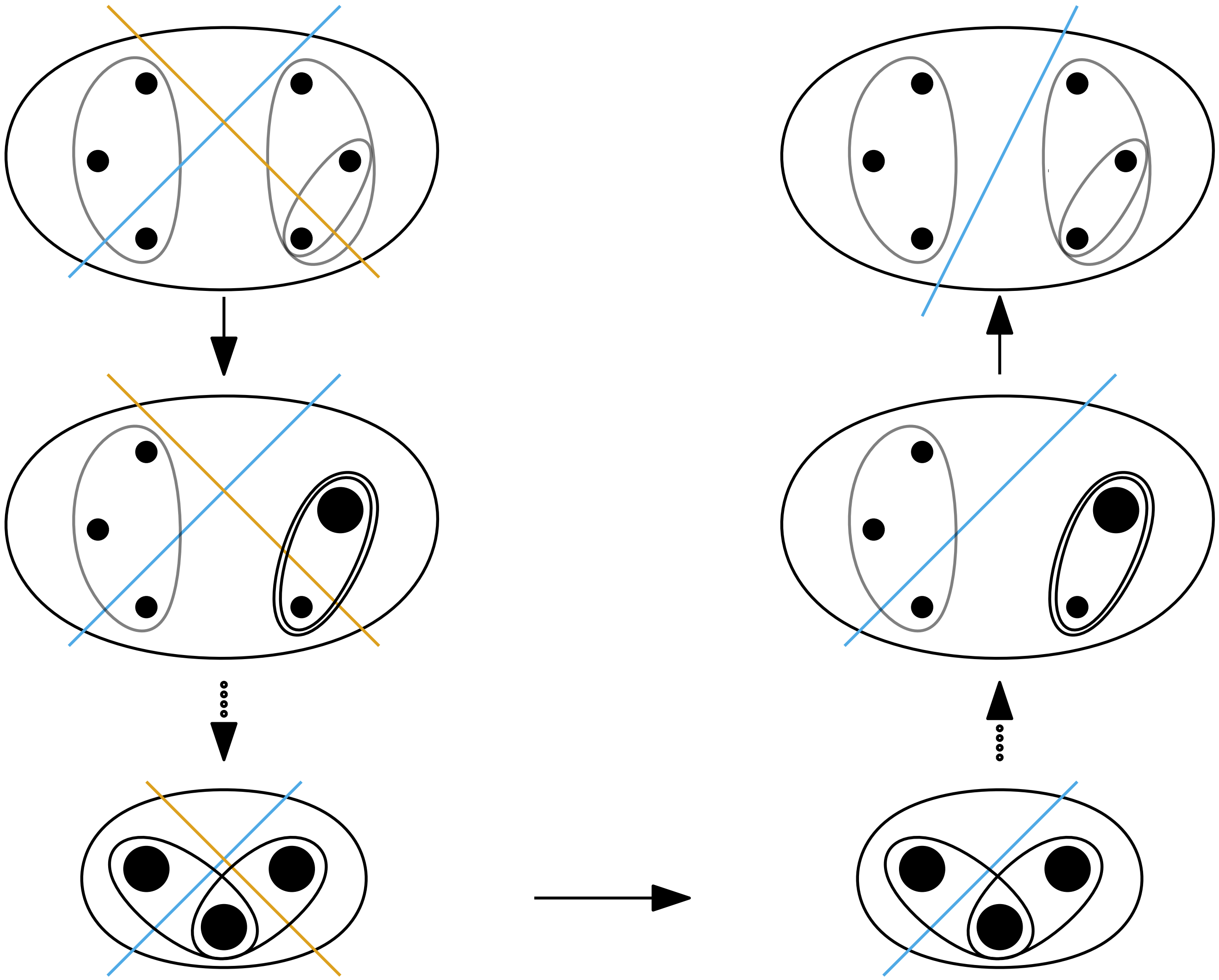


Initial Partitioning:

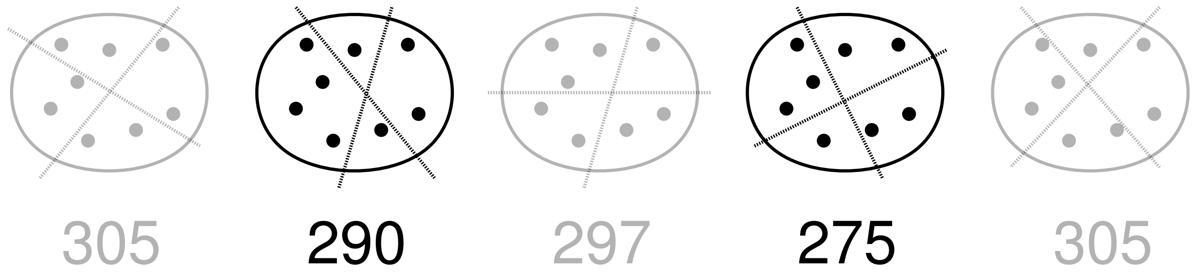
- Use the better parent partition (P_1)
- Maintains solution quality



Combine Operator

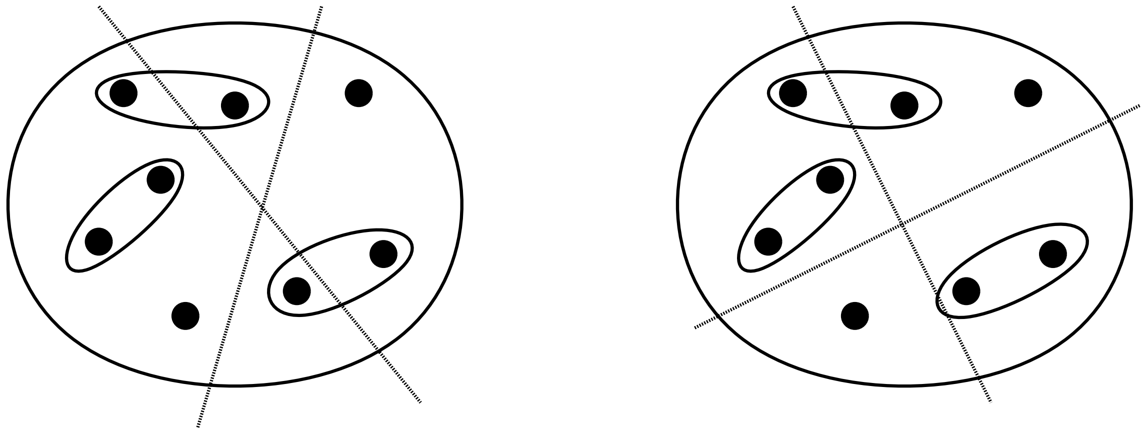


Edge Frequency Multicombine



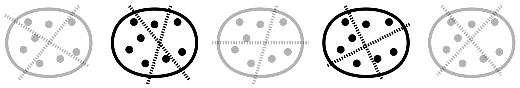
- We inspect the $\sqrt{|P|}$ best individuals of P
- Each hyperedge e is analyzed on its frequency in the selected elements

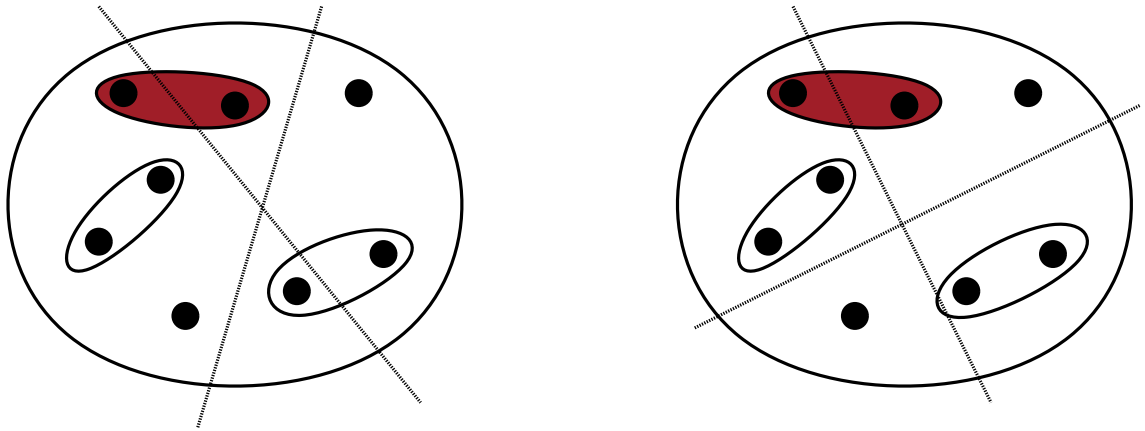
Edge Frequency Multicombine



- Frequent edges are beneficial to the solution quality
- Contracting frequent edges may be detrimental to solution quality
- Additionally it may limit other contractions

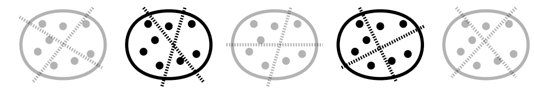
Edge Frequency Multicombine

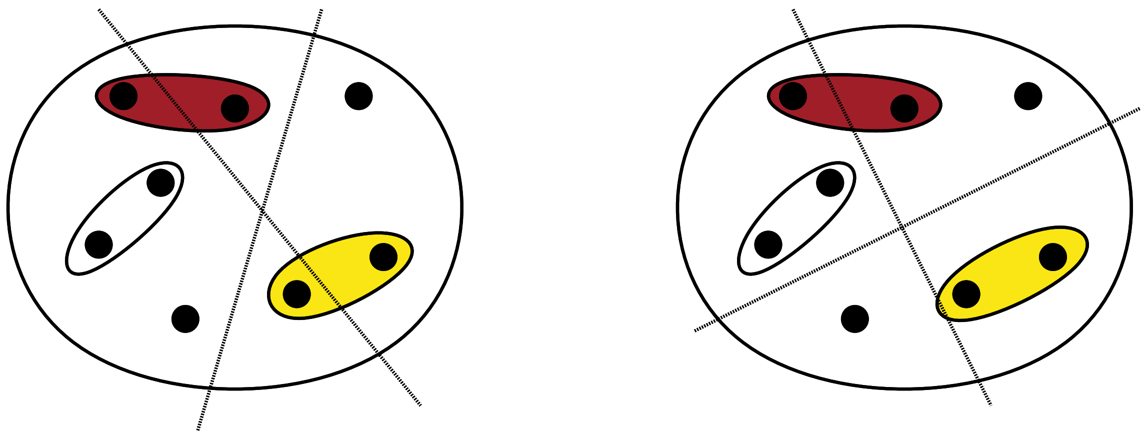




- Frequent edges are beneficial to the solution quality
- Contracting frequent edges may be detrimental to solution quality
- Additionally it may limit other contractions

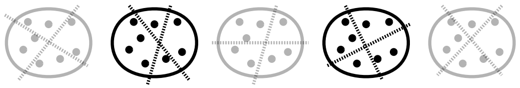
Edge Frequency Multicombine





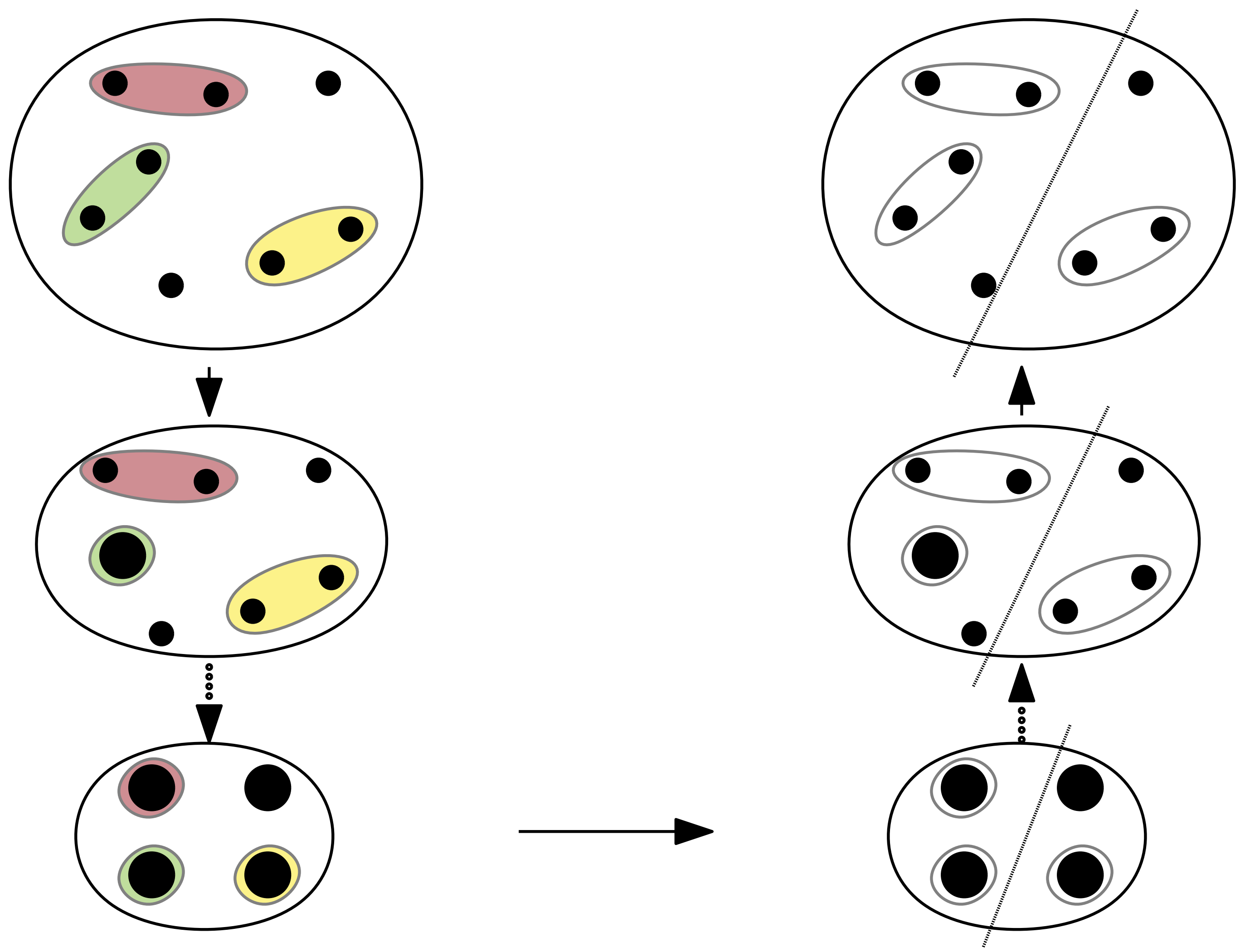
- Frequent edges are beneficial to the solution quality
- Contracting frequent edges may be detrimental to solution quality
- Additionally it may limit other contractions

Edge Frequency Multicombine

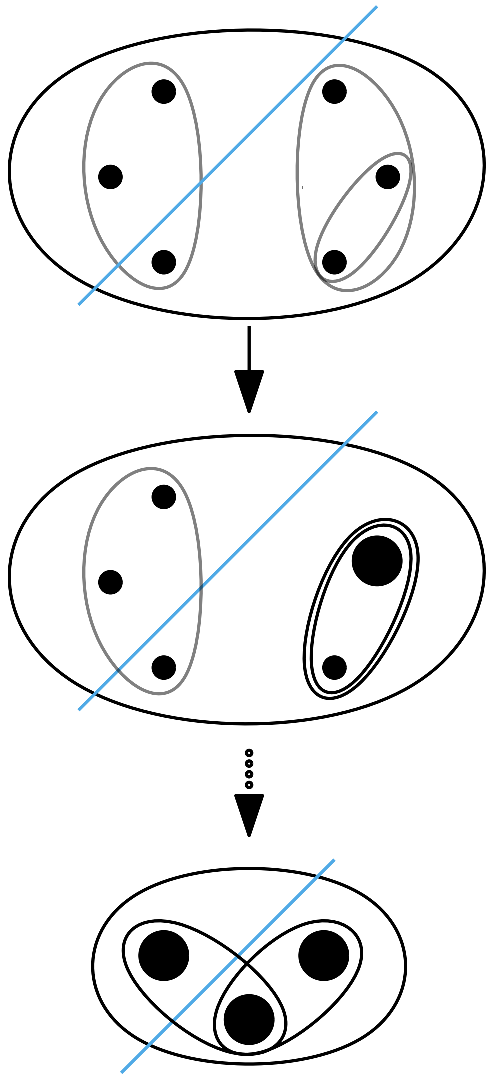


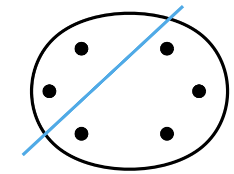
- Frequent edges are beneficial to the solution quality
- Contracting frequent edges may be detrimental to solution quality
- Additionally it may limit other contractions

Edge Frequency Multicombine



V-Cycle (+ New Initial Partitioning)



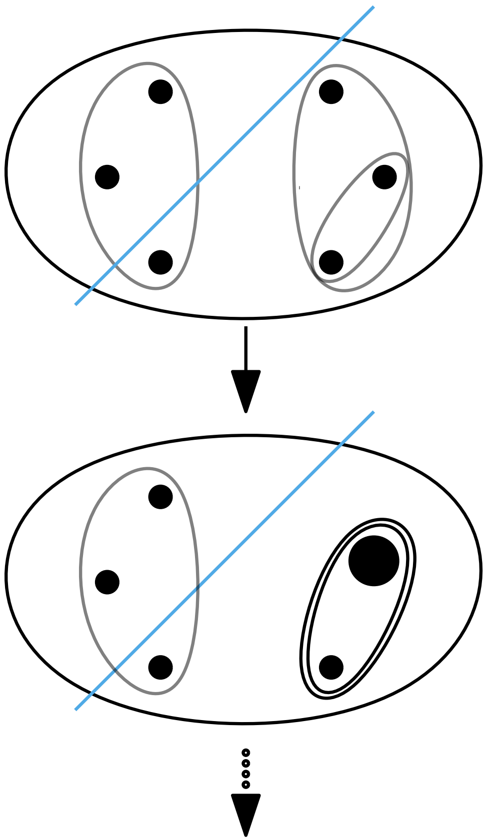


P

Coarsening:

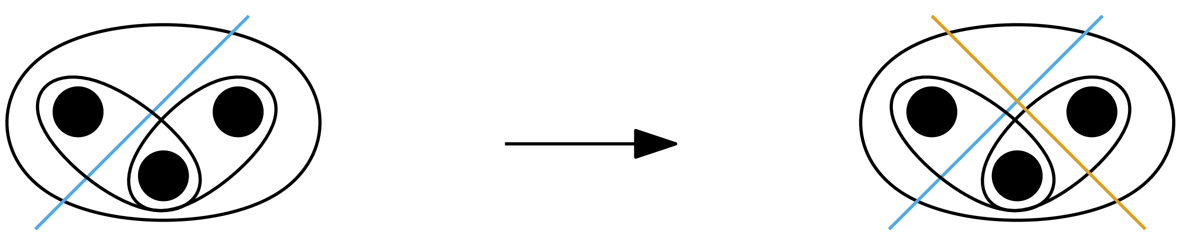
- Contractions must respect P
- Does not change solution quality

V-Cycle (+ New Initial Partitioning)

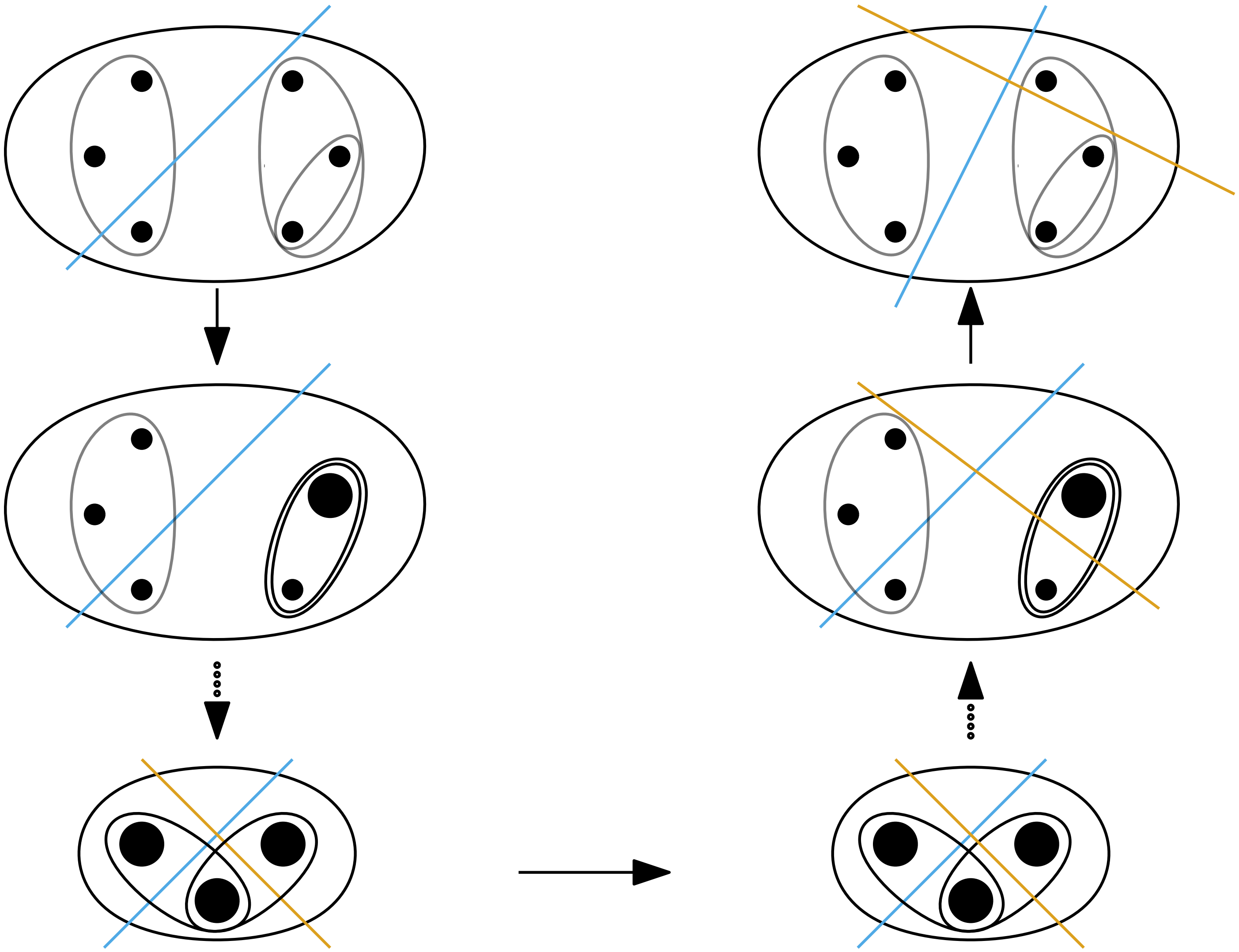


Initial Partitioning:

- V-Cycle can generate a new initial partitioning
- Or keep the current partition (maintains solution quality)



V-Cycle (+ New Initial Partitioning)



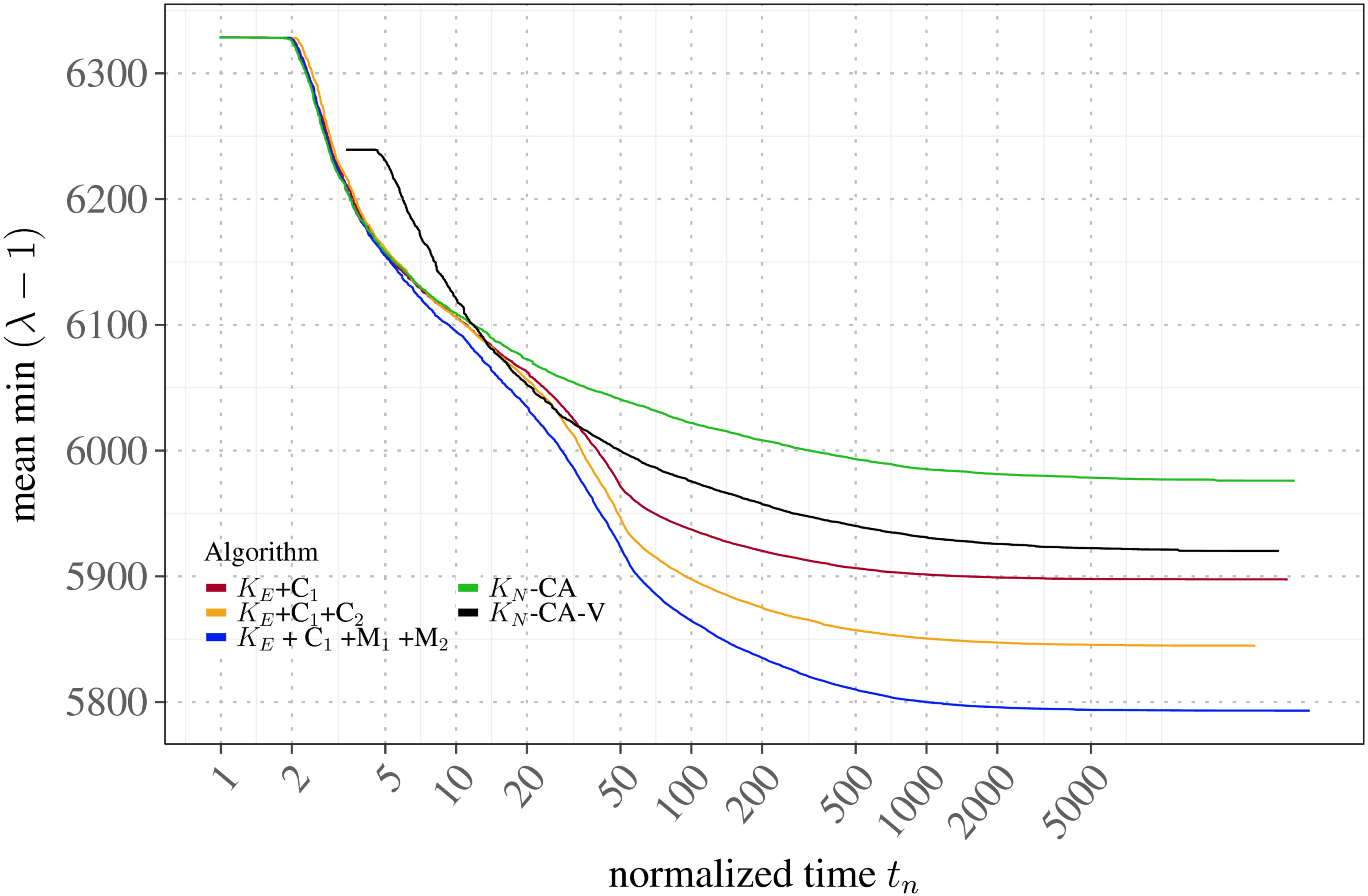
Experimental Setup

- $k = \{2, 4, 8, 16, 32, 64, 128\}; \epsilon = 0.03$
- 90 Hypergraph instances
- Comparison against repeated $KaHyPar$-CA

The run time is normalized $t_n = \frac{time}{t_1}$; t_1 := duration of first iteration.

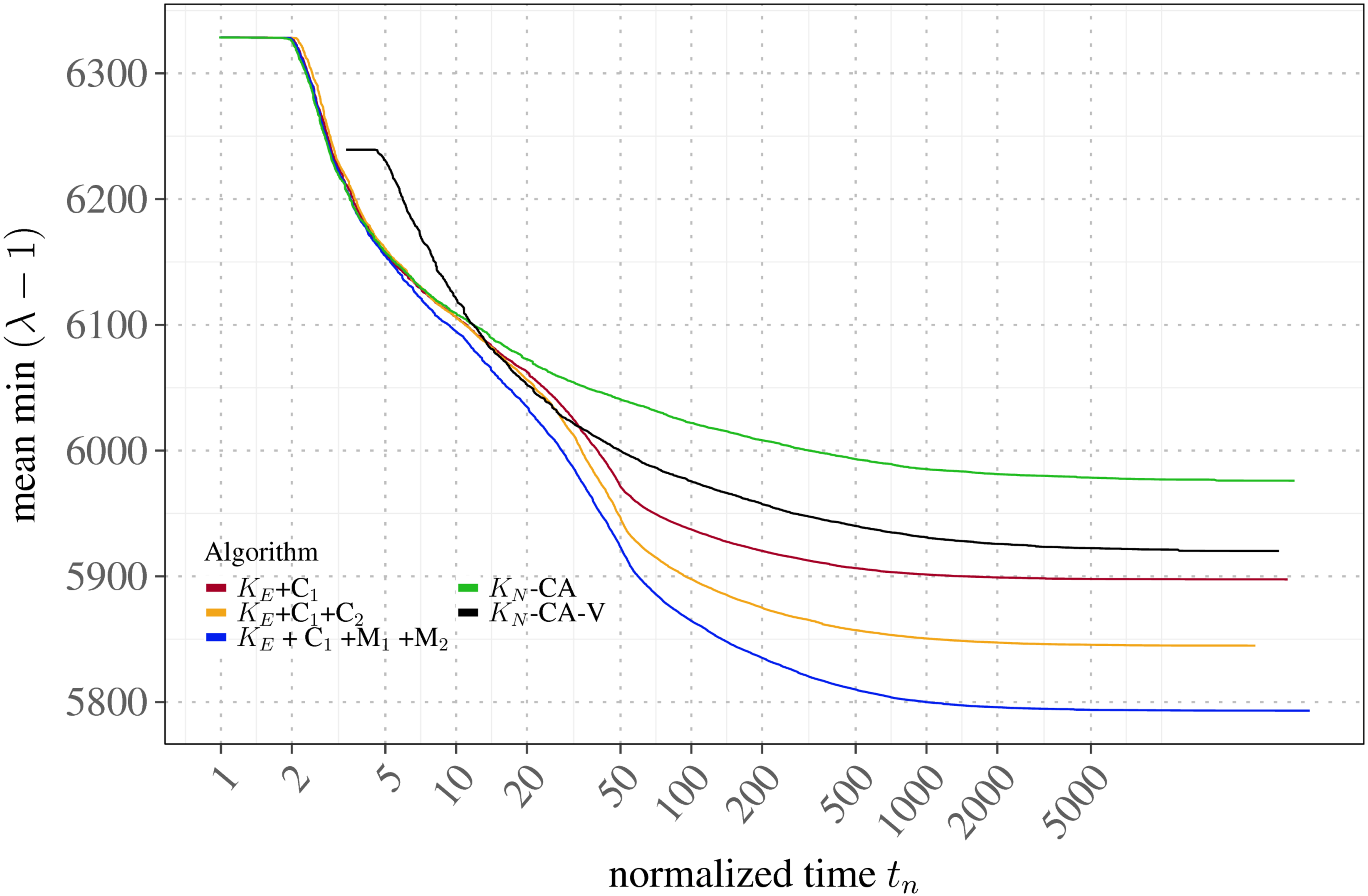
- Allows comparing differently sized hypergraphs
- Algorithmic components can be analyzed on run time

Results



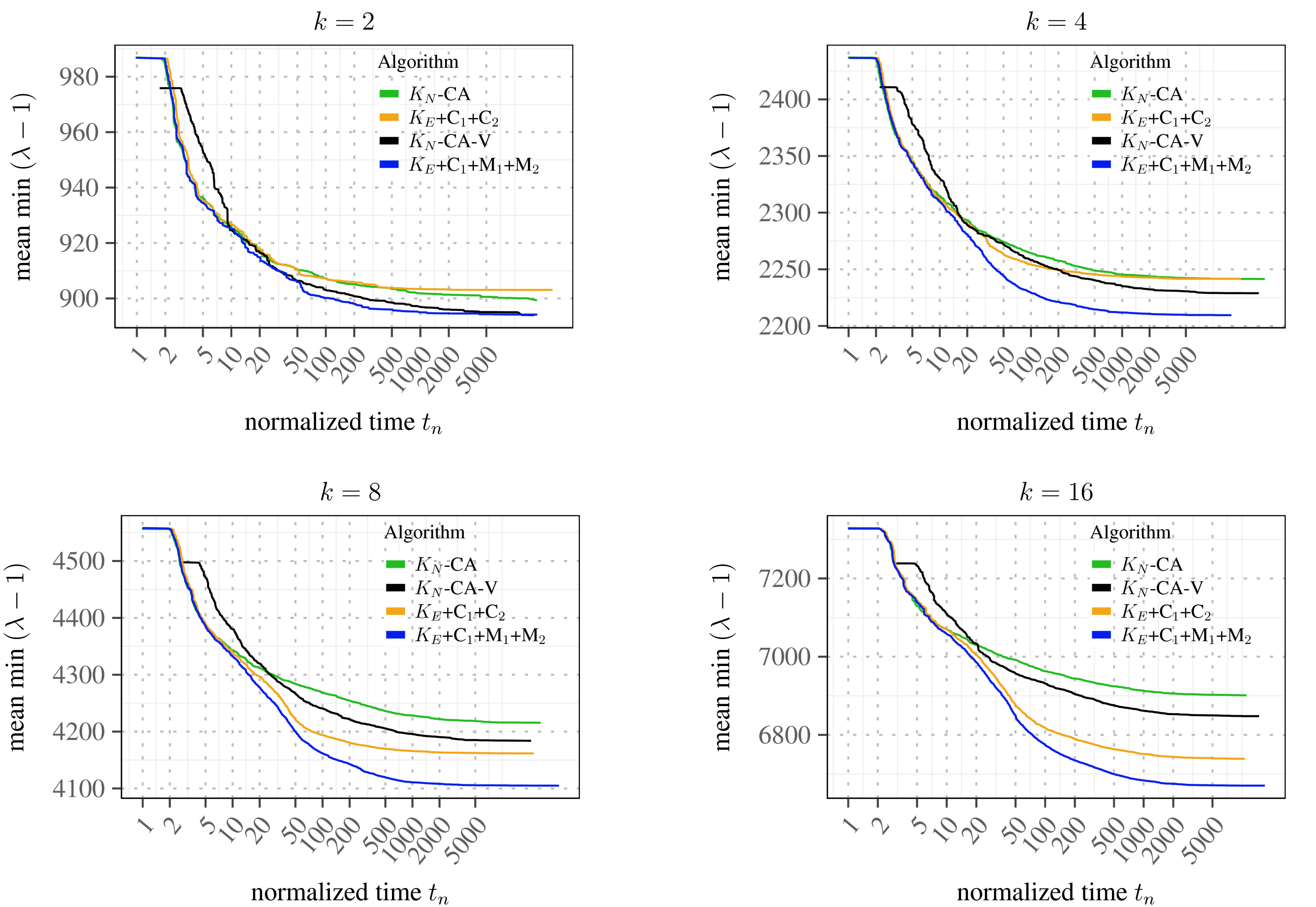
mean min $(\lambda - 1)$
6300
6200
6100
6000
5900
5800
normalized time t_n
1
2
5
10
20
50
100
200
500
1000
2000
5000
Algorithm
K_E+C_1
K_E+C_1+C_2
K_E + C_1 +M_1 +M_2
K_N-CA
K_N-CA-V

Results



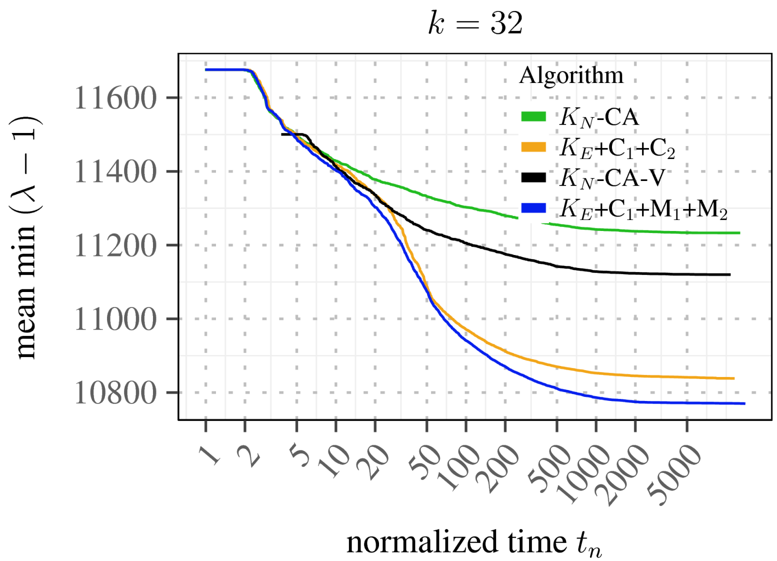
mean min $(\lambda - 1)$
6300
6200
6100
6000
5900
5800
normalized time t_n
1
2
5
10
20
50
100
200
500
1000
2000
5000
Algorithm
K_E+C_1
K_E+C_1+C_2
K_E + C_1 +M_1 +M_2
K_N-CA
K_N-CA-V

Results

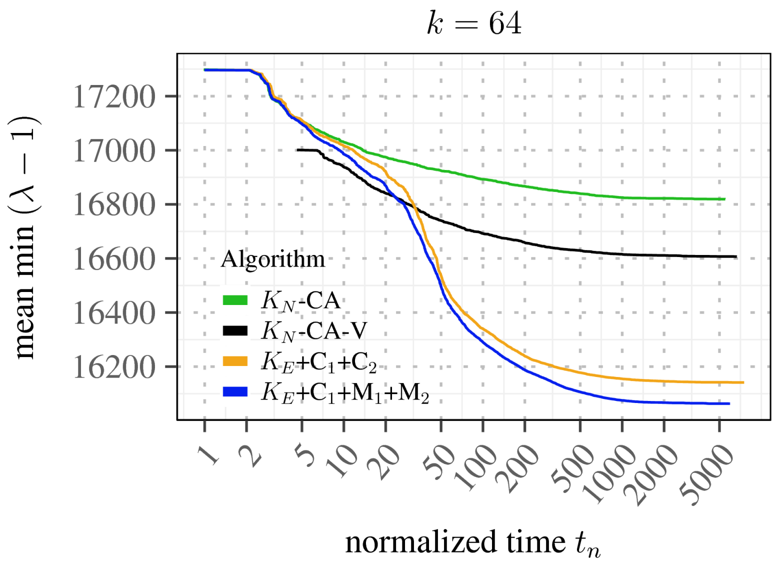


$k = 2$
$k = 4$
$k = 8$
$k = 16$
Algorithm
K_N-CA
K_E+C$_1$+C$_2$
K_N-CA-V
K_E+C$_1$+M$_1$+M$_2$
mean min $(\lambda - 1)$
normalized time t_n
1 2 5 10 20 50 100 200 500 1000 2000 5000
980 960 940 920 900
2400 2350 2300 2250 2200
4500 4400 4300 4200 4100
7200 7000 6800

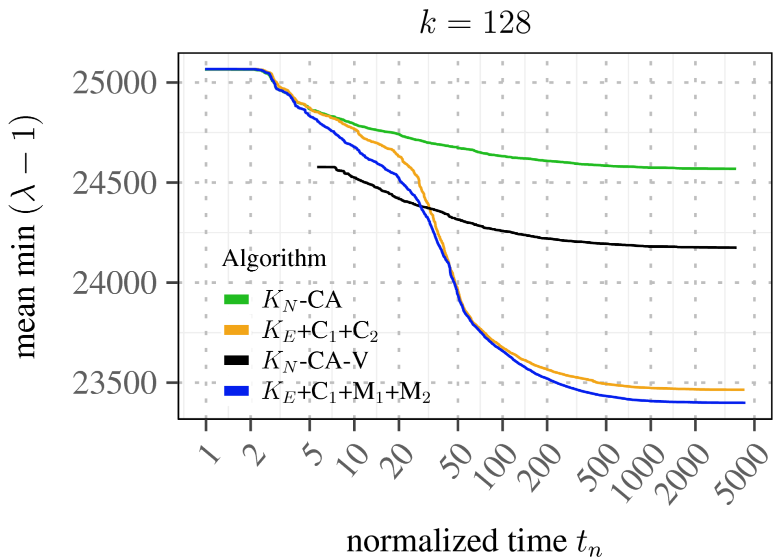
Results



$k = 32$
Algorithm
K_N-CA
K_E+C$_1$+C$_2$
K_N-CA-V
K_E+C$_1$+M$_1$+M$_2$
mean min ($\lambda - 1$)
11600
11400
11200
11000
10800
1
2
5
10
20
50
100
200
500
1000
2000
5000
normalized time t_n



$k = 64$
Algorithm
K_N-CA
K_N-CA-V
K_E+C$_1$+C$_2$
K_E+C$_1$+M$_1$+M$_2$
mean min ($\lambda - 1$)
17200
17000
16800
16600
16400
16200
1
2
5
10
20
50
100
200
500
1000
2000
5000
normalized time t_n



$k = 128$
Algorithm
K_N-CA
K_E+C$_1$+C$_2$
K_N-CA-V
K_E+C$_1$+M$_1$+M$_2$
mean min ($\lambda - 1$)
25000
24500
24000
23500
1
2
5
10
20
50
100
200
500
1000
2000
5000
normalized time t_n

Table of Improvements

Karlsruhe Institute of Technology

	K_E + C_1 + C_2		K_E + C_1 + M_1 + M_2	
k	K_N-CA-V	K_N-CA	K_N-CA-V	K_N-CA
all k	1.7%	2.7%	2.2%	3.2%
2	-0.2%	0.4%	0.2%	0.8%
4	-0.2%	0.3%	0.9%	1.3%
8	0.7%	1.6%	1.9%	2.7%
16	1.9%	2.8%	2.6%	3.5%
32	2.9%	3.9%	3.2%	4.2%
64	3.2%	4.7%	3.4%	4.8%
128	3.3%	5.0%	3.3%	5.0%

Conclusion

Conclusion

- $(\lambda - 1)$ improvement of up to 5%
- High integration of evolutionary aspects in the multilevel approach

Future Work

- Added parallelization for faster partitioning
- Different approach for generating the initial population
- Time cost analysis for evolutionary operators