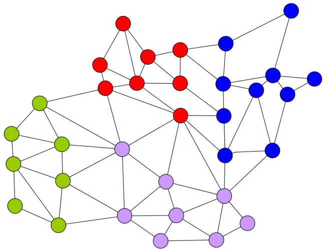
High Quality Graph Partitioning

Peter Sanders, Christian Schulz

Overview

- Introduction
- Multilevel Algorithms
- Advanced Techniques
- Evolutionary Techniques
- Experiments
- Summary

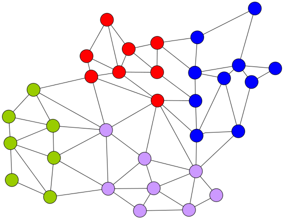


Department of Informatics
Institute for Theoretical Computer Science, Algorithmics II

ϵ-Balanced Graph Partitioning

Partition graph $G = (V, E, c : V \to \mathbf{R}_{>0}, \omega : E \to \mathbf{R}_{>0})$
into k disjoint blocks s.t.

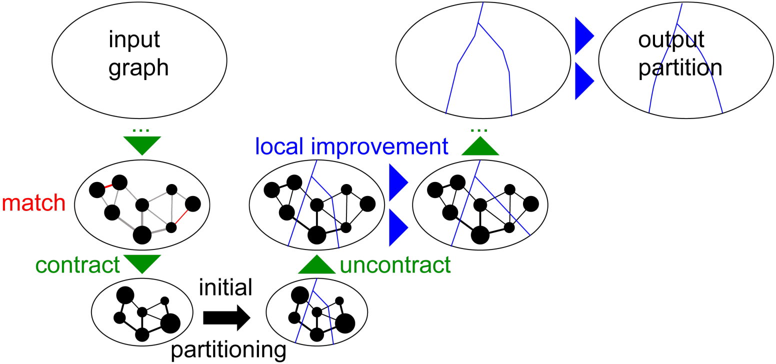
- total node weight of each block $\leq \dfrac{1+\epsilon}{k}$ total node weight
- total weight of cut edges as small as possible



Applications:
linear equation systems, VLSI design, route planning, ...

Department of Informatics
Institute for Theoretical Computer Science, Algorithmics II

Multi-Level Graph Partitioning



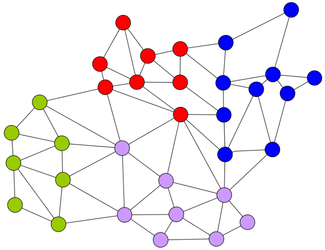
Successful in existing systems:
Metis, Scotch, Jostle,. . . , **KaPPa**, **KaSPar**, **KaFFPa**, **KaFFPaE**

Department of Informatics
Institute for Theoretical Computer Science, Algorithmics II

Advanced Techniques

Talk Today

- Edge Ratings
- High Quality Matchings
- Flow Based Refinements
- More Localized Local Search
- F-cycles for Graph Partitioning



Department of Informatics
Institute for Theoretical Computer Science, Algorithmics II

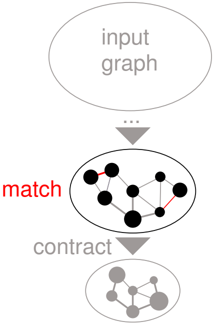
Graph Partitioning

Matching Selection

Goals:

1. large edge weights $\rightsquigarrow$ sparsify
2. large #edges $\rightsquigarrow$ few levels
3. uniform node weights $\rightsquigarrow$ “represent” input
4. small node degrees $\rightsquigarrow$ “represent” input

$\rightsquigarrow$ unclear objective
$\rightsquigarrow$ gap to approx. weighted matching
which only considers 1.,2.



Our Solution:
Apply approx. weighted matching to general edge rating function

 Peter Sanders, Christian Schulz:

Department of Informatics
Institute for Theoretical Computer Science, Algorithmics II

Graph Partitioning

Edge Ratings

$$\omega(\{u, v\})$$

$$\text{expansion}(\{u, v\}) := \frac{\omega(\{u, v\})}{c(u) + c(v)}$$

$$\text{expansion}^*(\{u, v\}) := \frac{\omega(\{u, v\})}{c(u)c(v)}$$

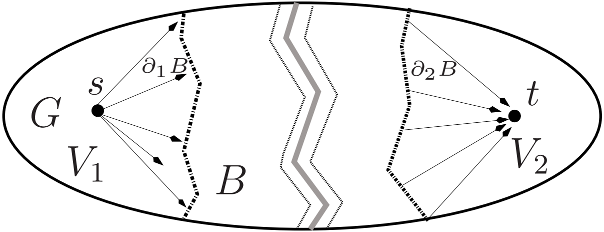
$$\text{expansion}^{*2}(\{u, v\}) := \frac{\omega(\{u, v\})^2}{c(u)c(v)}$$

$$\text{innerOuter}(\{u, v\}) := \frac{\omega(\{u, v\})}{\text{Out}(v) + \text{Out}(u) - 2\omega(u, v)}$$

where $c =$ node weight, $\omega =$edge weight,
$\text{Out}(u) := \sum_{\{u,v\} \in E} \omega(\{u, v\})$

Flows as Local Improvement

Two Blocks

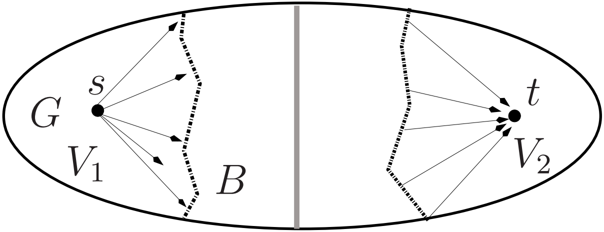


- area B, such that each (s, t)-min cut is ϵ-balanced cut in G
- e.g. 2 times BFS (left, right)
- stop the BFS, if size would exceed $(1+\epsilon)\frac{c(V)}{2} - c(V_2)$
- $\Rightarrow c(V_{2_{\text{new}}}) \leq c(V_2) + (1+\epsilon)\frac{c(V)}{2} - c(V_2)$

Department of Informatics
Institute for Theoretical Computer Science, Algorithmics II

Flows as Local Improvement

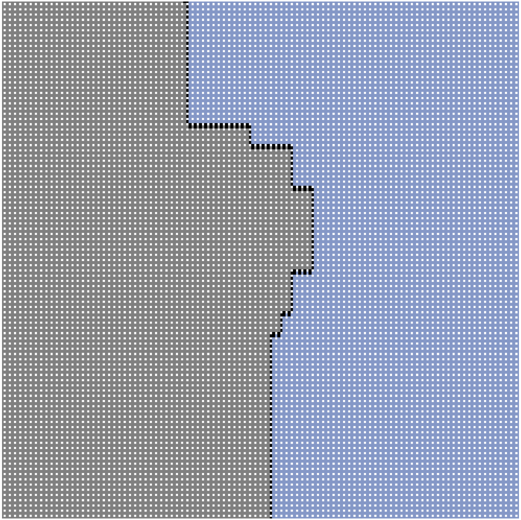
Two Blocks



- obtain optimal cut in B
- since each cut in B yields a feasible partition
  $\rightarrow$ improved two-partition
- advanced techniques possible and necessary

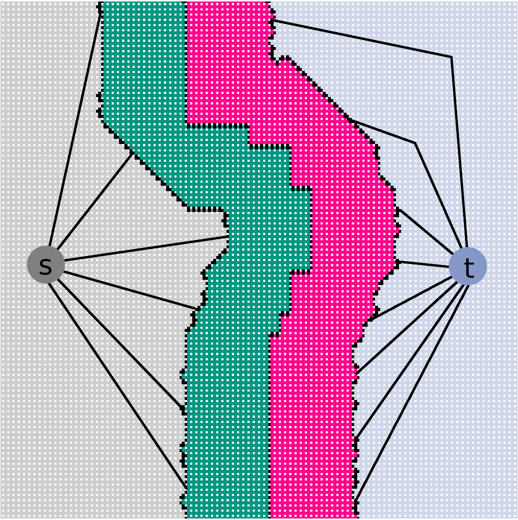
Example

100x100 Grid



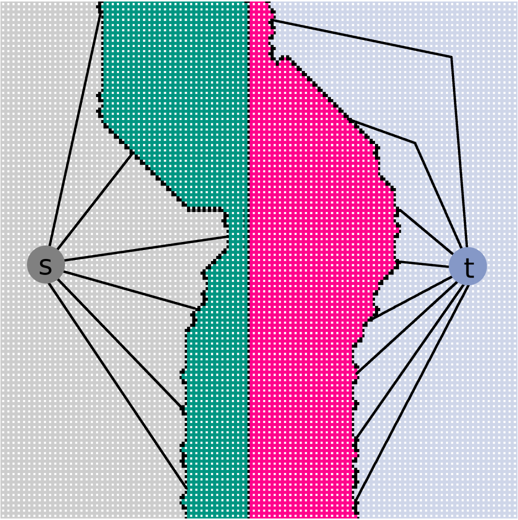
Example

Constructed Flow Problem (using BFS)



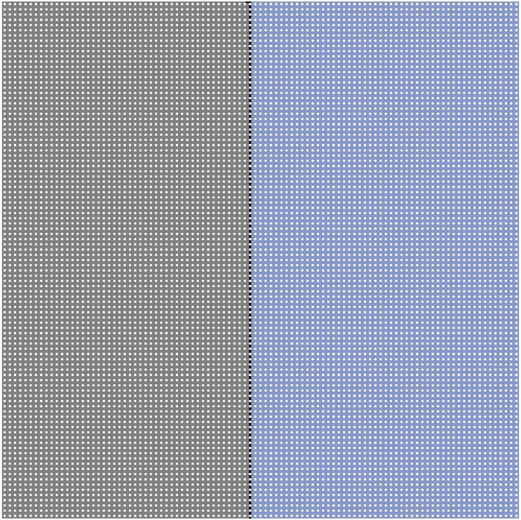
Example

Apply Max-Flow Min-Cut



Example

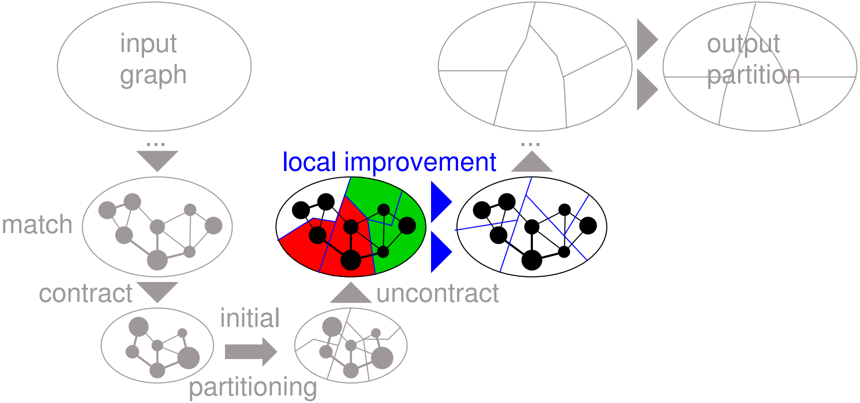
Output Improved Partition



Local Improvement for k-partitions

Using Flows?

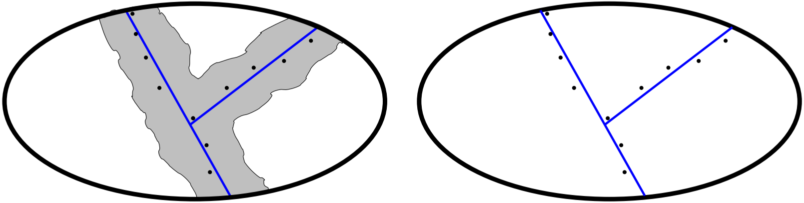
on each pair of blocks



Department of Informatics
Institute for Theoretical Computer Science, Algorithmics II

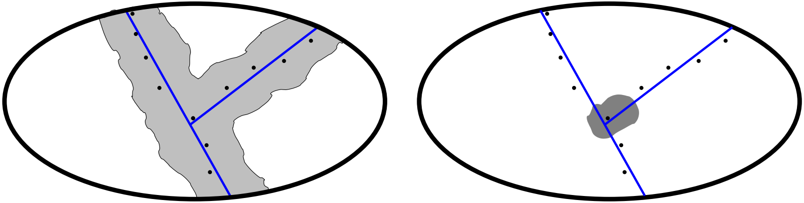
More Localized Local Search

- Idea: *KaPPa*, *KaSPar* $\Rightarrow$ more local searches are better
- Typical: k-way local search initialized with complete boundary
- Localization:
  1. complete boundary $\Rightarrow$ maintained todo list T
  2. initialize search with single node $v \in_{\text{rnd}} T$
  3. iterate until $T = \emptyset$
- each node moved at most once



More Localized Local Search

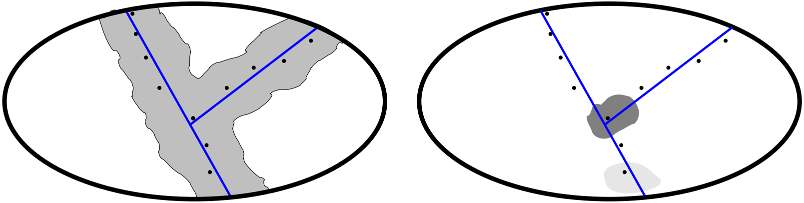
- Idea: *KaPPa*, *KaSPar* $\Rightarrow$ more local searches are better
- Typical: k-way local search initialized with complete boundary
- Localization:
  1. complete boundary $\Rightarrow$ maintained todo list T
  2. initialize search with single node $v \in_{\text{rnd}} T$
  3. iterate until $T = \emptyset$
- each node moved at most once



Department of Informatics
Institute for Theoretical Computer Science, Algorithmics II

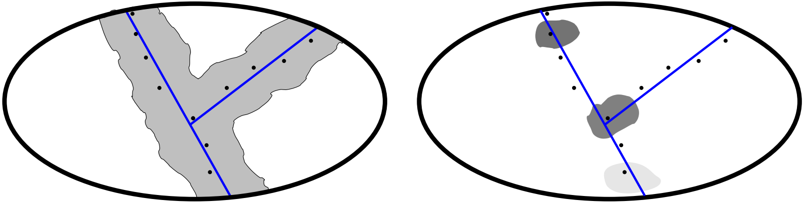
More Localized Local Search

- Idea: *KaPPa*, *KaSPar* $\Rightarrow$ more local searches are better
- Typical: k-way local search initialized with complete boundary
- Localization:
  1. complete boundary $\Rightarrow$ maintained todo list T
  2. initialize search with single node $v \in_{\text{rnd}} T$
  3. iterate until $T = \emptyset$
- each node moved at most once



More Localized Local Search

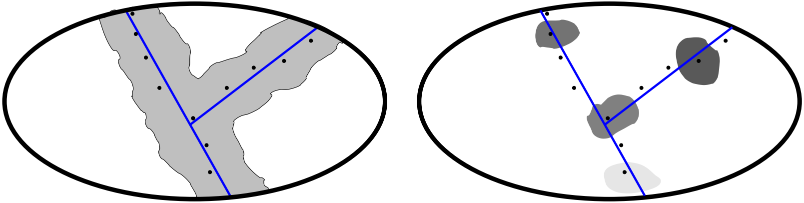
- Idea: *KaPPa*, *KaSPar* $\Rightarrow$ more local searches are better
- Typical: k-way local search initialized with complete boundary
- Localization:
  1. complete boundary $\Rightarrow$ maintained todo list T
  2. initialize search with single node $v \in_{\text{rnd}} T$
  3. iterate until $T = \emptyset$
- each node moved at most once



Department of Informatics
Institute for Theoretical Computer Science, Algorithmics II

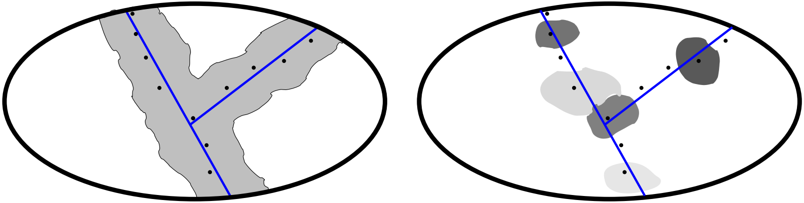
More Localized Local Search

- Idea: *KaPPa*, *KaSPar* $\Rightarrow$ more local searches are better
- Typical: k-way local search initialized with complete boundary
- Localization:
  1. complete boundary $\Rightarrow$ maintained todo list T
  2. initialize search with single node $v \in_{\text{rnd}} T$
  3. iterate until $T = \emptyset$
- each node moved at most once



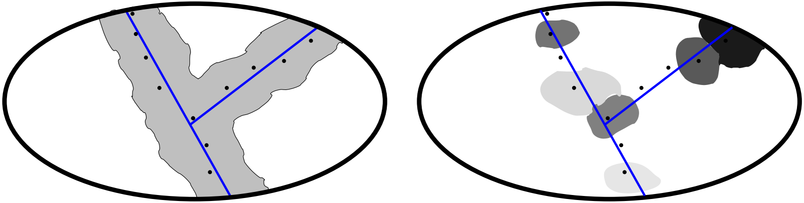
More Localized Local Search

- Idea: *KaPPa*, *KaSPar* $\Rightarrow$ more local searches are better
- Typical: k-way local search initialized with complete boundary
- Localization:
  1. complete boundary $\Rightarrow$ maintained todo list T
  2. initialize search with single node $v \in_{\text{rnd}} T$
  3. iterate until $T = \emptyset$
- each node moved at most once



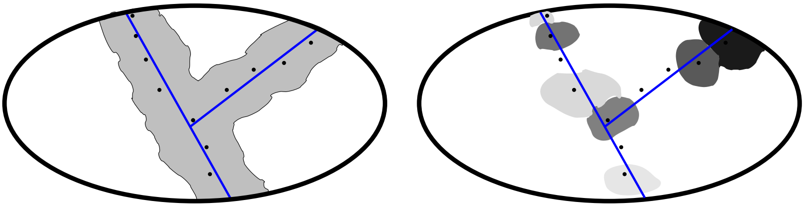
More Localized Local Search

- Idea: *KaPPa*, *KaSPar* $\Rightarrow$ more local searches are better
- Typical: k-way local search initialized with complete boundary
- Localization:
  1. complete boundary $\Rightarrow$ maintained todo list T
  2. initialize search with single node $v \in_{\text{rnd}} T$
  3. iterate until $T = \emptyset$
- each node moved at most once



More Localized Local Search

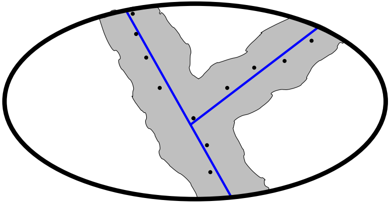
- Idea: *KaPPa*, *KaSPar* $\Rightarrow$ more local searches are better
- Typical: k-way local search initialized with complete boundary
- Localization:
  1. complete boundary $\Rightarrow$ maintained todo list T
  2. initialize search with single node $v \in_{\text{rnd}} T$
  3. iterate until $T = \emptyset$
- each node moved at most once



Department of Informatics
Institute for Theoretical Computer Science, Algorithmics II

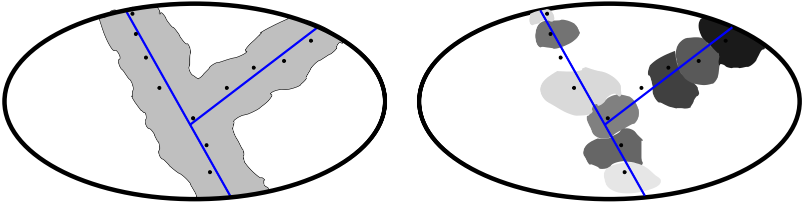
More Localized Local Search

- Idea: *KaPPa*, *KaSPar* $\Rightarrow$ more local searches are better
- Typical: k-way local search initialized with complete boundary
- Localization:
  1. complete boundary $\Rightarrow$ maintained todo list T
  2. initialize search with single node $v \in_{\text{rnd}} T$
  3. iterate until $T = \emptyset$
- each node moved at most once



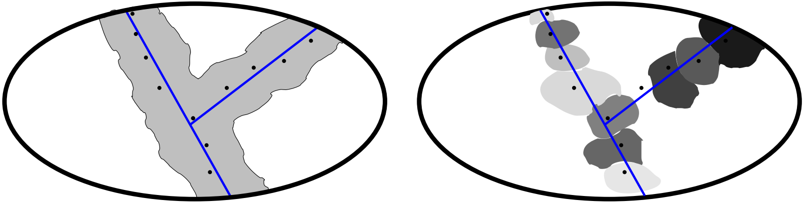
More Localized Local Search

- Idea: *KaPPa*, *KaSPar* $\Rightarrow$ more local searches are better
- Typical: k-way local search initialized with complete boundary
- Localization:
  1. complete boundary $\Rightarrow$ maintained todo list T
  2. initialize search with single node $v \in_{\text{rnd}} T$
  3. iterate until $T = \emptyset$
- each node moved at most once



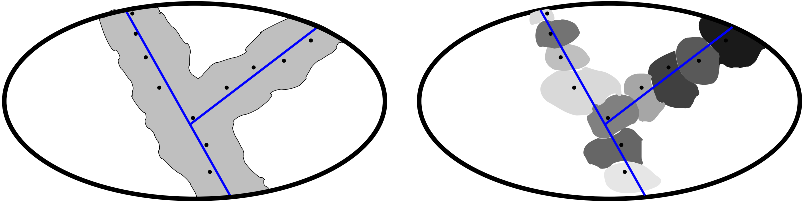
More Localized Local Search

- Idea: *KaPPa*, *KaSPar* $\Rightarrow$ more local searches are better
- Typical: k-way local search initialized with complete boundary
- Localization:
  1. complete boundary $\Rightarrow$ maintained todo list T
  2. initialize search with single node $v \in_{\text{rnd}} T$
  3. iterate until $T = \emptyset$
- each node moved at most once



More Localized Local Search

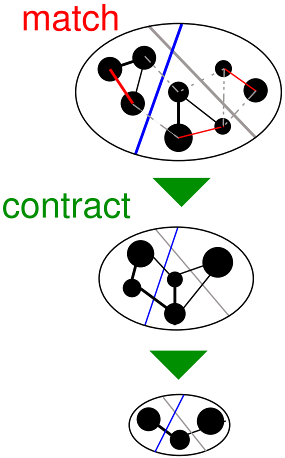
- Idea: *KaPPa*, *KaSPar* $\Rightarrow$ more local searches are better
- Typical: k-way local search initialized with complete boundary
- Localization:
  1. complete boundary $\Rightarrow$ maintained todo list T
  2. initialize search with single node $v \in_{\text{rnd}} T$
  3. iterate until $T = \emptyset$
- each node moved at most once



Distributed
Evolutionary Graph Partitioning

- Evolutionary Algorithms:
  - highly inspired by biology
  - population of individuals
  - selection, mutation, recombination, ...
- Goal: Integrate KaFFPa in an Evolutionary Strategy
- Evolutionary Graph Partitioning:
  individuals $\leftrightarrow$ partitions
  fitness $\leftrightarrow$ edge cut
- Parallelization $\rightarrow$ quality records in a few minutes for small graphs

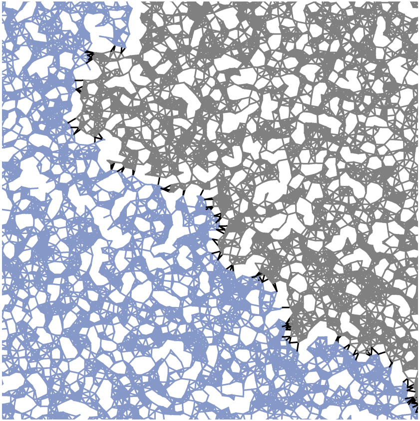
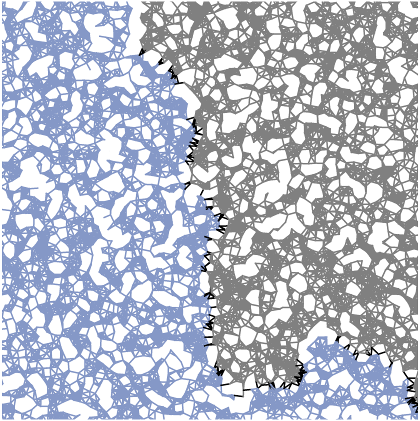
Combine



- two individuals $\mathcal{P}_1$, $\mathcal{P}_2$:
  don't contract cut edges of $\mathcal{P}_1$ or $\mathcal{P}_2$
- until no matchable edge is left
- coarsest graph $\leftrightarrow$ Q-graph of overlay
- $\rightarrow$ exchanging good parts is easy
- inital solution: use better of both parents

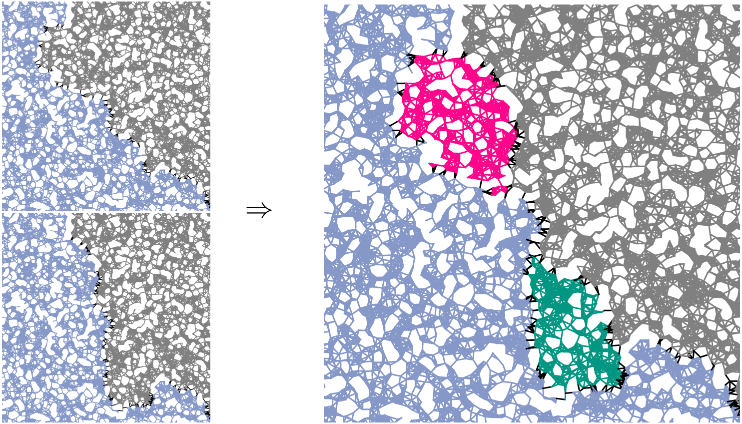
Example

Two Individuals $\mathcal{P}_1$, $\mathcal{P}_2$

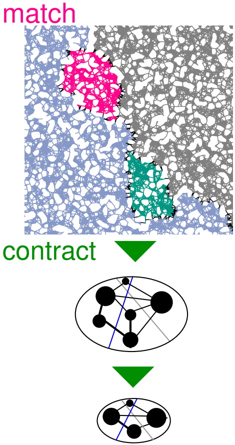
Example

Overlay of $\mathcal{P}_1$, $\mathcal{P}_2$

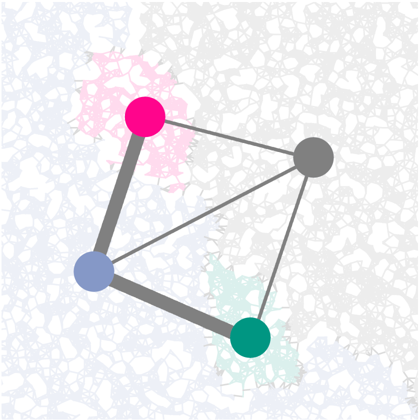


Example

Multilevel Combine of $\mathcal{P}_1$, $\mathcal{P}_2$



$\Rightarrow$

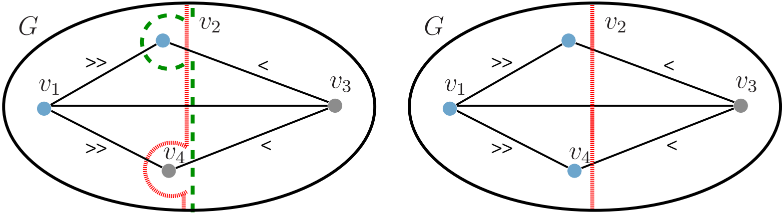


Peter Sanders, Christian Schulz:
High Quality Graph Partitioning

Department of Informatics
Institute for Theoretical Computer Science, Algorithmics II

Exchanging good parts is easy

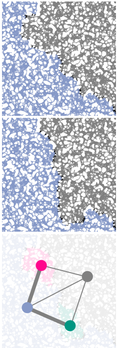
Coarsest Level



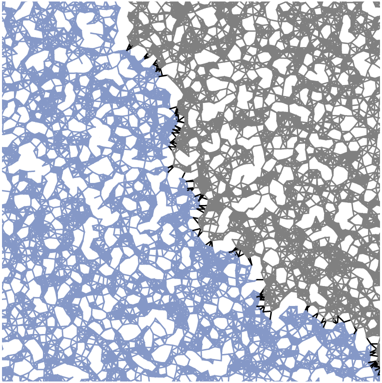
- $>>$ large weight, $<$ small weight
- start with the better partition (red, $\mathcal{P}_2$)
- move v_4 to the opposite block
- integrated into multilevel scheme (+local search on each level)

Example

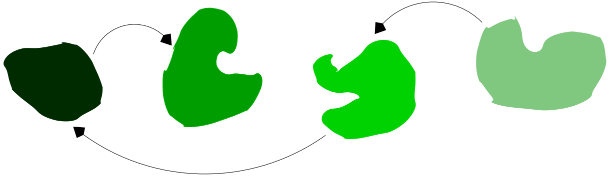
Result of $\mathcal{P}_1$, $\mathcal{P}_2$



$\Rightarrow$



Parallelization



- each PE has its own island (a local population)
- locally: perform combine and mutation operations
- communicate analog to *randomized rumor spreading*
    1. rumor $\leftrightarrow$ currently best local partition
    2. local best partition *changed* $\rightarrow$ send it to $\mathcal{O}(\log P)$ random PEs
    3. asynchronous communication (MPI Isend)

Department of Informatics
Institute for Theoretical Computer Science, Algorithmics II

Experimental Results

Comparison with Other Systems

Geometric mean, imbalance $\epsilon = 0.03$:
11 graphs (78K–18M nodes) $\times k \in \{2, 4, 8, 16, 64\}$

Algorithm	large graphs		
	Best	Avg.	t[s]
KaFFPa strong	12 053	12 182	121.22
KaSPar strong	12 450	+3%	87.12
KaFFPa eco	12 763	+6%	3.82
Scotch	14 218	+20%	3.55
KaFFa fast	15 124	+24%	0.98
kMetis	15 167	+33%	0.83

- Repeating Scotch as long as KaSPar strong run and choosing the best result $\rightsquigarrow$ 12.1% larger cuts
- Walshaw instances, road networks, Florida Sparse Matrix Collection, random Delaunay triangulations, random geometric graphs

Quality

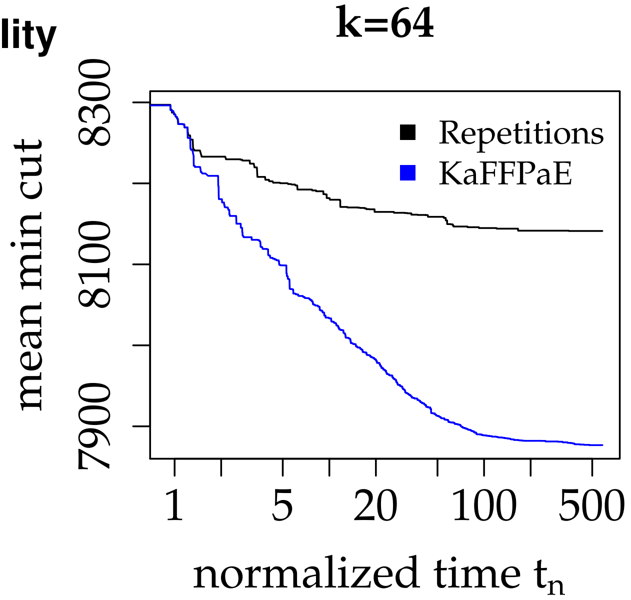
Evolutionary Graph Partitioning

blocks k	KaFFPaE improvement over reps. of KaFFPa
2	0.2%
4	1.0%
8	1.5%
16	2.7%
32	3.4%
64	3.3%
128	3.9%
256	3.7%
overall	2.5%

2h time, 32 cores per graph and k, geom. mean

Department of Informatics
Institute for Theoretical Computer Science, Algorithmics II

Quality


k=64
Repetitions
KaFFPaE
mean min cut
7900
8100
8300
1
5
20
100
500
normalized time t_n

Walshaw Benchmark

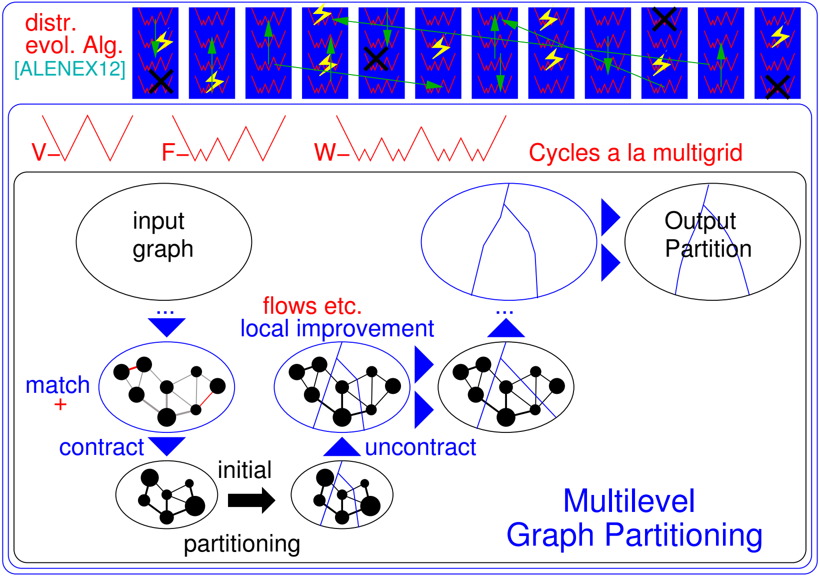
- runtime is not an issue
- 614 instances ($\epsilon \in \{1\%, 3\%, 5\%\}$)
- focus on partition quality

Algorithm	$<$	$\leq$
KaPPa	131	189
KaSPar	155	238
KaFFPa	317	435
KaFFPaE	300	470

- overall quality records $\leq$:

ϵ	$\leq$
1%	78%
3%	92%
5%	94%

Summary



 Peter Sanders, Christian Schulz:

Department of Informatics
Institute for Theoretical Computer Science, Algorithmics II

Outlook

- **Further Material in the Paper(s)**
  - F-cycles, High Quality Matchings, ...
  - Different combine and mutation operators
  - Specialization to road networks (Buffoon)
  - Many more details and experiments ...
- **Future Work**
  - other objective functions
    - currently via selection criterion
    - connectivity? $\tilde{f}(\mathcal{P}) := f(\mathcal{P}) + \chi_{\{\mathcal{P} \text{ not connected}\}} \cdot |E|$
  - integrate other partitioners
  - graph clustering
  - open source release

Department of Informatics
Institute for Theoretical Computer Science, Algorithmics II

Thank you!

Contact: christian.schulz@kit.edu
sanders@kit.edu