Partitioning into Colorful Components: From Parameterized Algorithmics to Algorithm Engineering

Rolf Niedermeier[1]

[1]Institut für Softwaretechnik und Theoretische Informatik,
TU Berlin

8 July 2012

Colorful Components

Colorful Components

Instance: An undirected graph $G = (V, E)$ and a coloring of the vertices. **Task:** Delete a minimum number of edges such that all connected components are **colorful**, that is, they do not contain two vertices of the same color.

Colorful Components

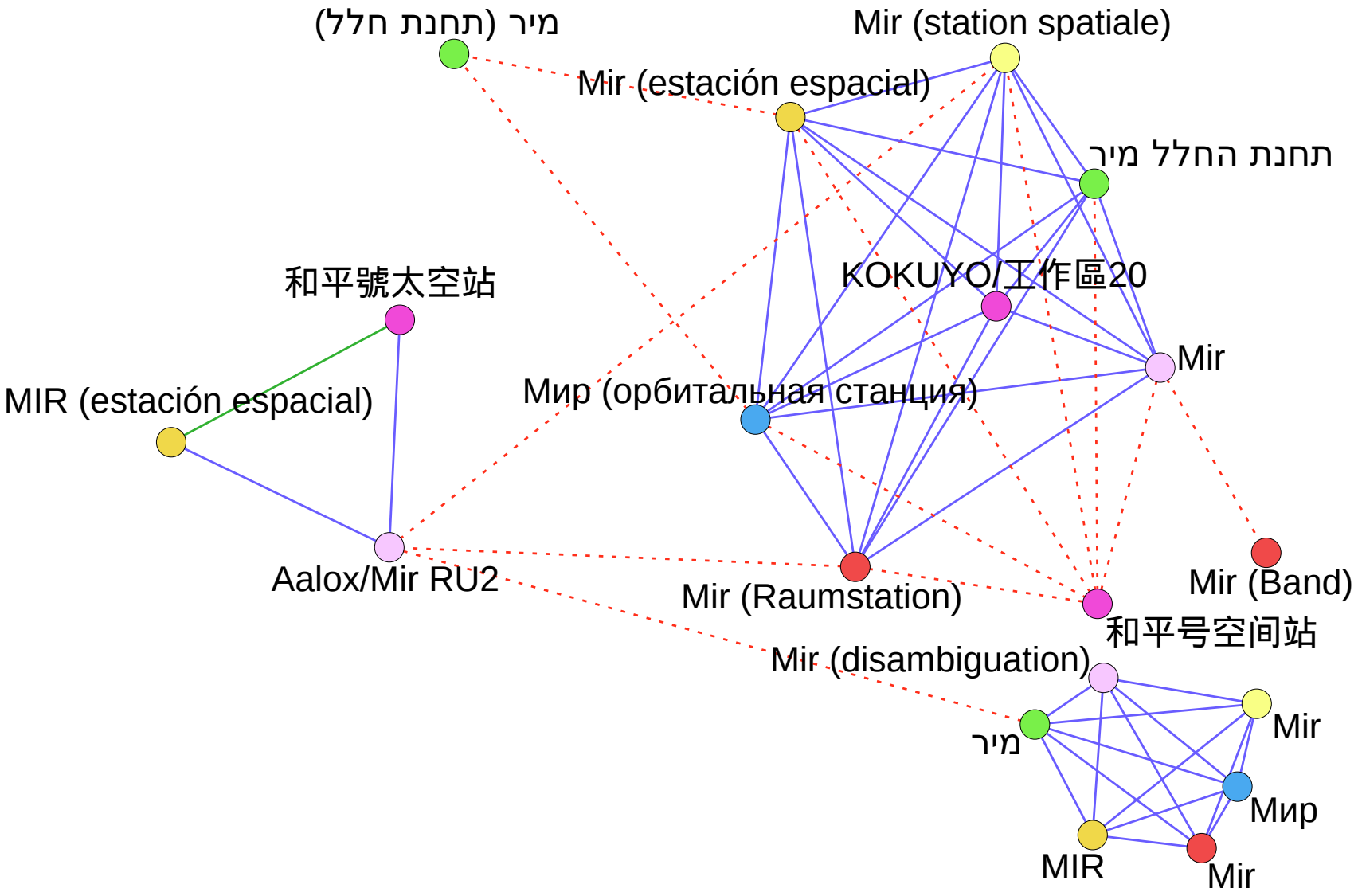
Colorful Components

Instance: An undirected graph $G = (V, E)$ and a coloring of the vertices. **Task:** Delete a minimum number of edges such that all connected components are **colorful**, that is, they do not contain two vertices of the same color.

Motivation / Applications:

- Part of a Multiple Sequence Alignment Pipeline suggested by Corel, Pitschi, and Morgenstern (Bioinformatics 2010).
- Network Alignment for multiple protein-protein interaction networks.
- Correcting interlanguage links in Wikipedia.
- Matching entries in different databases (e.g. online social networks or matching products in online stores).

Example from Wikipedia interlanguage link correction



Related, more general problems

Colorful Components is a special case of

- **Multicut:** For a given set of vertex pairs in a graph, find a minimum number of edges to delete such that each pair gets disconnected.

Related, more general problems

Colorful Components is a special case of

- **Multicut:** For a given set of vertex pairs in a graph, find a minimum number of edges to delete such that each pair gets disconnected.
  $\rightsquigarrow$ To transform a Colorful Components instance into a Multicut instance, set the vertex pairs to be all vertices of same color which are connected by a path.

Related, more general problems

Colorful Components is a special case of

- **Multicut:** For a given set of vertex pairs in a graph, find a minimum number of edges to delete such that each pair gets disconnected.
  $\rightsquigarrow$ To transform a Colorful Components instance into a Multicut instance, set the vertex pairs to be all vertices of same color which are connected by a path.
- **Multi-Multiway Cut:** For given vertex subsets in a graph, find a minimum number of edges to delete such that no pair of vertices within a subset lies in the same connected component of the resulting graph.

Related, more general problems

Colorful Components is a special case of

- **Multicut:** For a given set of vertex pairs in a graph, find a minimum number of edges to delete such that each pair gets disconnected.
  $\rightsquigarrow$ To transform a Colorful Components instance into a Multicut instance, set the vertex pairs to be all vertices of same color which are connected by a path.
- **Multi-Multiway Cut:** For given vertex subsets in a graph, find a minimum number of edges to delete such that no pair of vertices within a subset lies in the same connected component of the resulting graph.
  $\rightsquigarrow$ To transform a Colorful Components instance into a Multi-Multiway Cut instance, set the vertex subsets to be the color classes.

More on Wikipedia interlanguage link correction I

Labyrinthulomycetes – Wikipedia, the free encyclopedia – Iceweasel

File Edit View History Bookmarks Tools Help

W Labyrinthulomycetes - Wikipedi...

en.wikipedia.org/wiki/Labyrinthulomycetes

Google

Log in / create account

WIKIPEDIA
The Free Encyclopedia

Article Talk Read Edit View history Search

Labyrinthulomycetes

From Wikipedia, the free encyclopedia

The **Labyrinthulomycetes** (ICBN) or **Labyrinthulea**[1] (ICZN), or **Slime nets** are a class of protists that produce a network of filaments or tubes,[2] which serve as tracks for the cells to glide along and absorb nutrients for them. There are two main groups, the labyrinthulids and thraustochytrids. They are mostly marine, commonly found as parasites on alga and seagrass or as decomposers on dead plant material. They also include some parasites of marine invertebrates.

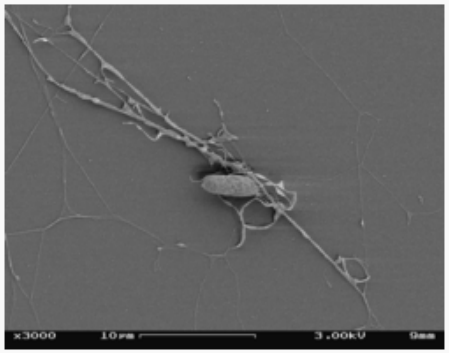
Although they are outside the cells, the filaments are surrounded by a membrane. They are formed and connected with the cytoplasm by a unique organelle called a sagenogen or bothrosome. The cells are uninucleate and typically ovoid, and move back and forth along the amorphous network at speeds varying from 5-150 µm per minute. Among the labyrinthulids the cells are enclosed within the tubes, and among the thraustochytrids they are attached to their sides.

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction
Toolbox
Print/export

Languages
Česky
Deutsch
Español
日本語
Македонски
Norsk (bokmål)

Slime nets



The cell with the network of filaments *Aplanochytrium sp.*

Scientific classification

Domain: Eukaryota
Kingdom: Chromalveolata
Phylum: Heterokontophyta
Class: **Labyrinthulomycetes** DICK, 2001 or

More on Wikipedia interlanguage link correction I

Labyrinthulomycetes – Wikipedia, the free encyclopedia – Iceweasel

File Edit View History Bookmarks Tools Help

W Labyrinthulomycetes - Wikipedi...

en.wikipedia.org/wiki/Labyrinthulomycetes

Google

Log in / create account

WIKIPEDIA
The Free Encyclopedia

Article Talk Read Edit View history Search

Labyrinthulomycetes

From Wikipedia, the free encyclopedia

The **Labyrinthulomycetes** (ICBN) or **Labyrinthulea**[1] (ICZN), or **Slime nets** are a class of protists that produce a network of filaments or tubes,[2] which serve as tracks for the cells to glide along and absorb nutrients for them. There are two main groups, the labyrinthulids and thraustochytrids. They are mostly marine, commonly found as parasites on alga and seagrass or as decomposers on dead plant material. They also include some parasites of marine invertebrates.

Although they are outside the cells, the filaments are

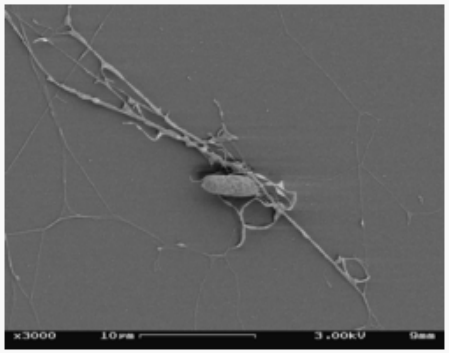
attached to their sides.

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction
Toolbox
Print/export

Languages
Česky
Deutsch
Español
日本語
Македонски
Norsk (bokmål)

Slime nets



The cell with the network of filaments *Aplanochytrium sp.*

Scientific classification

Domain: Eukaryota
Kingdom: Chromalveolata
Phylum: Heterokontophyta
Class: **Labyrinthulomycetes** DICK, 2001 or

Netzschleimpilze

Die **Netzschleimpilze** oder **Schleimnetze** (Labyrinthulomycetes) bilden ein Taxon innerhalb der Stramenopilen und sind somit näher mit Braunalgen, Goldalgen

More on Wikipedia interlanguage link correction II

Example scenario:
Wrong interlanguage links:
Schinken (German) $\rightarrow$ Prosciutto (Italian) $\rightarrow$ ...(Russian) $\rightarrow$
Parmaschinken (German)

More on Wikipedia interlanguage link correction II

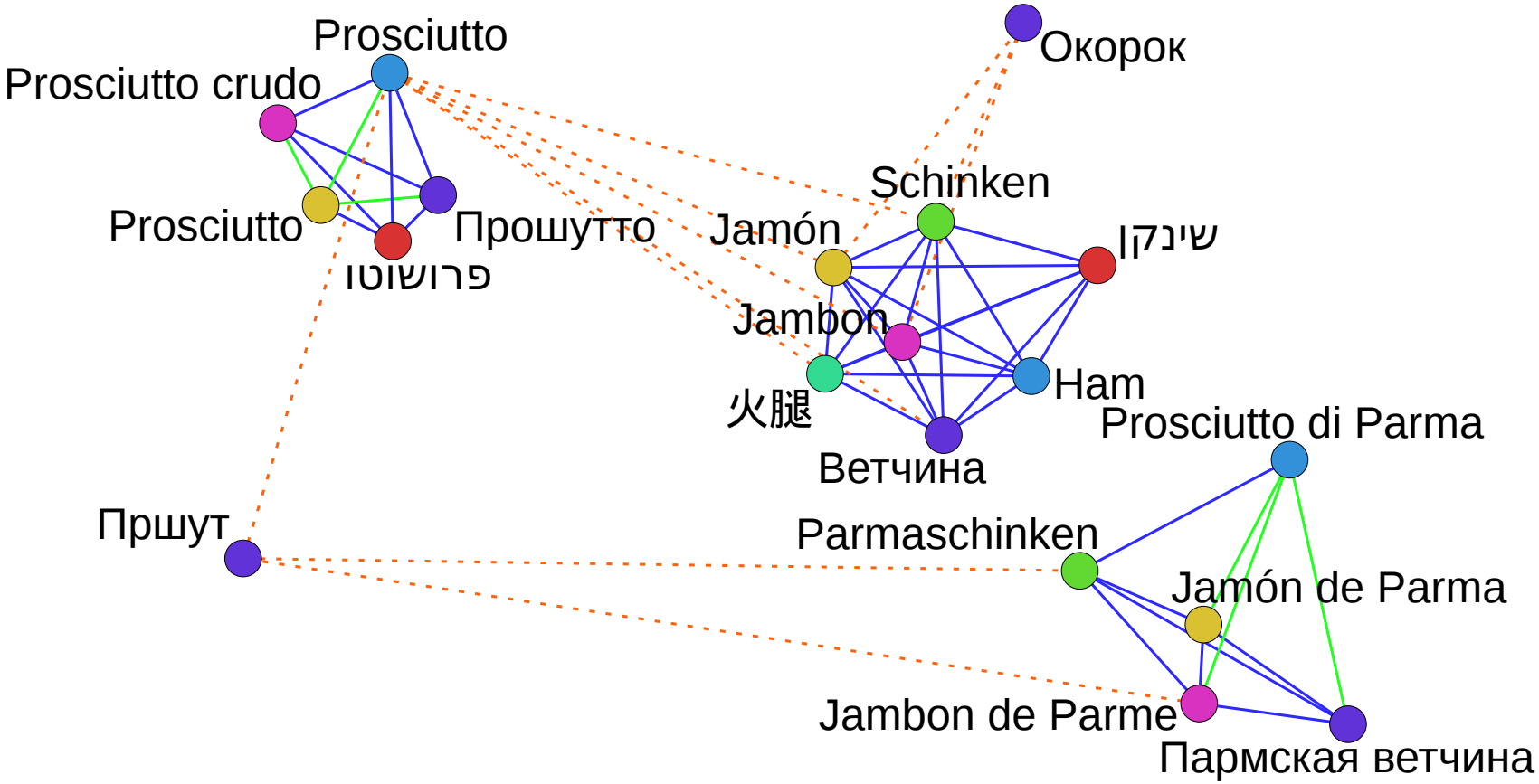
Example scenario:
Wrong interlanguage links:
Schinken (German) $\rightarrow$ Prosciutto (Italian) $\rightarrow$ ...(Russian) $\rightarrow$ Parmaschinken (German)

Assumption

If there is a link path from a word in some language to a different word in the same language, then at least one of the links on the path is wrong.

$\rightsquigarrow$ Fix inconsistencies ("in a maximum parsiomony way") using Colorful Components.

More on Wikipedia interlanguage link correction III



(Parameterized) Complexity Analysis

Facts:

- Colorful Components is fixed-parameter tractable with respect to the number of edge deletions, since Multicut is (Bousquet/Daligault/Thomassé ; Marz/Razgon, STOC 2011). However, running time $2^{O(k^3)} \cdot n^{O(1)}$ is impractical!

(Parameterized) Complexity Analysis

Facts:

- Colorful Components is fixed-parameter tractable with respect to the number of edge deletions, since Multicut is (Bousquet/Daligault/Thomassé ; Marz/Razgon, STOC 2011). However, running time $2^{O(k^3)} \cdot n^{O(1)}$ is impractical!
- Since there is also a parameterized reduction from Multicut to Colorful Components, there is little hope for efficient fixed-parameter tractability without considering fuerther parameters.

(Parameterized) Complexity Analysis

Facts:

- Colorful Components is fixed-parameter tractable with respect to the number of edge deletions, since Multicut is (Bousquet/Daligault/Thomassé ; Marz/Razgon, STOC 2011). However, running time $2^{O(k^3)} \cdot n^{O(1)}$ is impractical!
- Since there is also a parameterized reduction from Multicut to Colorful Components, there is little hope for efficient fixed-parameter tractability without considering fuerther parameters.

Two further candidates for parameterization:

- Tree-likeness? No, since NP-hard on trees!

(Parameterized) Complexity Analysis

Facts:

- Colorful Components is fixed-parameter tractable with respect to the number of edge deletions, since Multicut is (Bousquet/Daligault/Thomassé ; Marz/Razgon, STOC 2011). However, running time $2^{O(k^3)} \cdot n^{O(1)}$ is impractical!
- Since there is also a parameterized reduction from Multicut to Colorful Components, there is little hope for efficient fixed-parameter tractability without considering fuerther parameters.

Two further candidates for parameterization:

- Tree-likeness? No, since NP-hard on trees!
- Number of colors? No, since NP-hard for three colors!

NP-hardness and fixed-parameter tractability on trees

Proposition

Colorful Components is NP-hard on diameter-four trees.

Simple reduction from Multicut in stars: For every vertex pair $\{s, t\}$ to be disconnected, insert two new degree-one vertices of same unique color and attach them to s and t, respectively. Each original vertex gets a further unique color.

NP-hardness and fixed-parameter tractability on trees

Proposition

Colorful Components is NP-hard on diameter-four trees.

Simple reduction from Multicut in stars: For every vertex pair $\{s, t\}$ to be disconnected, insert two new degree-one vertices of same unique color and attach them to s and t, respectively. Each original vertex gets a further unique color.

Good news:

Proposition

Colorful Components can be solved in $2^c \cdot n^{O(1)}$ time on trees.

Idea of proof: Dynamic programming.

NP-hardness and fixed-parameter tractability on trees

Proposition

Colorful Components is NP-hard on diameter-four trees.

Simple reduction from Multicut in stars: For every vertex pair $\{s, t\}$ to be disconnected, insert two new degree-one vertices of same unique color and attach them to s and t, respectively. Each original vertex gets a further unique color.

Good news:

Proposition

Colorful Components can be solved in $2^c \cdot n^{O(1)}$ time on trees.

Idea of proof: Dynamic programming.

Observation: Number of colors small in several applications, but where do we encounter only trees...?

NP-hardness and fixed-parameter tractability on trees

Proposition

Colorful Components is NP-hard on diameter-four trees.

Simple reduction from Multicut in stars: For every vertex pair $\{s, t\}$ to be disconnected, insert two new degree-one vertices of same unique color and attach them to s and t, respectively. Each original vertex gets a further unique color.

Good news:

Proposition

Colorful Components can be solved in $2^c \cdot n^{O(1)}$ time on trees.

Idea of proof: Dynamic programming.

Observation: Number of colors small in several applications, but where do we encounter only trees...? Don't know!

Parameter number k of colors

Proposition

Colorful Components in two-colored graphs can solved in $O(\sqrt{n}m)$ time.

Idea of proof: Reduce to matching in bipartite graphs.

Parameter number k of colors

Proposition

Colorful Components in two-colored graphs can solved in $O(\sqrt{n}m)$ time.

Idea of proof: Reduce to matching in bipartite graphs.

Theorem

Colorful Components is NP-hard on three-colored graphs with maximum degree six; moreover, unless the ETH fails, it cannot be solved in

- $2^{o(k)} \cdot n^{O(1)}$ time;
- $2^{o(n)} \cdot n^{O(1)}$ time;
- $2^{o(m)} \cdot n^{O(1)}$ time;

Idea of proof: Reduction from 3-SAT (gadgeteering required!).

Combined parameter “deleted edges” and “colors”

Theorem

For $c \geqslant 3$ colors, Colorful Components can be solved in $O((c-1)^k \cdot m)$ time.

Idea of proof: Search tree:

Simple: Size-c^k search tree: If there are two vertices of same color in the same connected component, then there must be such a pair connected by a length-at-most-c path.

$\rightsquigarrow$ branch on one of edges of this path.

Combined parameter "deleted edges" and "colors"

Theorem

For $c \geqslant 3$ colors, Colorful Components can be solved in $O((c-1)^k \cdot m)$ time.

Idea of proof: Search tree:

Simple: Size-c^k search tree: If there are two vertices of same color in the same connected component, then there must be such a pair connected by a length-at-most-c path.
$\rightsquigarrow$ branch on one of edges of this path.

Less simple: More clever strategy to branch on only $c-1$ edges by exploiting degree-three vertices on the path (case distinction)...

Combined parameter "deleted edges" and "colors"

Theorem

For $c \geqslant 3$ colors, Colorful Components can be solved in $O((c-1)^k \cdot m)$ time.

Idea of proof: Search tree:

Simple: Size-c^k search tree: If there are two vertices of same color in the same connected component, then there must be such a pair connected by a length-at-most-c path.
$\rightsquigarrow$ branch on one of edges of this path.

Less simple: More clever strategy to branch on only $c-1$ edges by exploiting degree-three vertices on the path (case distinction)...

Corollary

Colorful Components has a factor $c-1$ approximation.

Data reduction rules

Trivial Rule. If a connected component contains no color more than once, then remove it.

$\rightsquigarrow$ Exhaustive application yields $2kc$-vertex problem kernel, which can be further improved to $(1+\epsilon)kc$ vertices...

Data reduction rules

Trivial Rule. If a connected component contains no color more than once, then remove it.

$\rightsquigarrow$ Exhaustive application yields $2kc$-vertex problem kernel, which can be further improved to $(1+\epsilon)kc$ vertices...

Edge Cut Rule. Let B be a minimal edge cut and G_B one side of the edge cut and let N be vertices incident to B-edges but not in G_B.

Data reduction rules

Trivial Rule. If a connected component contains no color more than once, then remove it.

$\rightsquigarrow$ Exhaustive application yields $2kc$-vertex problem kernel, which can be further improved to $(1+\epsilon)kc$ vertices...

Edge Cut Rule. Let B be a minimal edge cut and G_B one side of the edge cut and let N be vertices incident to B-edges but not in G_B.
If G_B contains no two vertices of the same color and if G_B is $|B|$-edge connected and each color of N-vertices occurs in G_B, then delete B and decrease k by $|B|$.

Data reduction rules

Trivial Rule. If a connected component contains no color more than once, then remove it.

$\rightsquigarrow$ Exhaustive application yields $2kc$-vertex problem kernel, which can be further improved to $(1+\epsilon)kc$ vertices...

Edge Cut Rule. Let B be a minimal edge cut and G_B one side of the edge cut and let N be vertices incident to B-edges but not in G_B.
If G_B contains no two vertices of the same color and if G_B is $|B|$-edge connected and each color of N-vertices occurs in G_B, then delete B and decrease k by $|B|$.

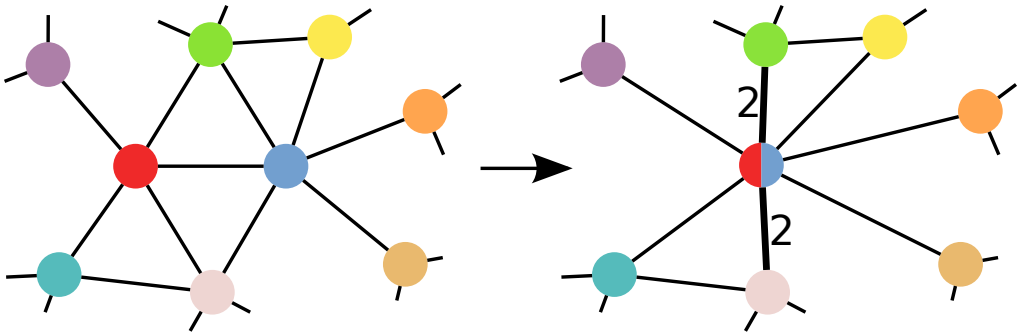
Open: Polynomial-time executability for nonconstant $|B|$.

Practical combinatorial algorithms: vertex merging

Idea

If we know that two vertices must belong to the same colorful component, then we want to be able to simplify the instance by merging them.

$\rightsquigarrow$ **Approach:** To merge vertices, introduce color *sets* per vertex and edge weights:



Exact edge branching algorithm

Edge branching

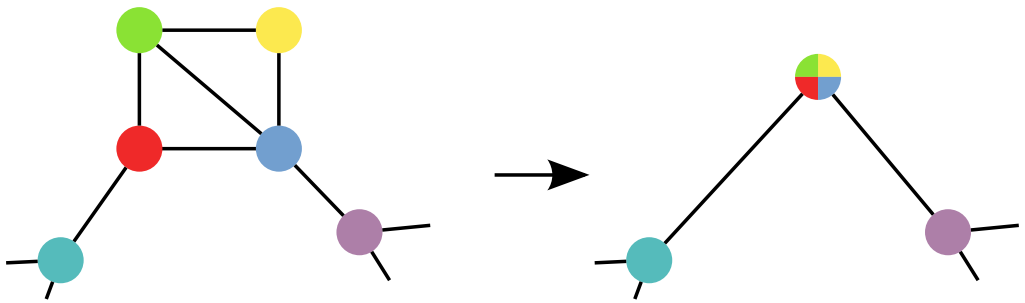
Branch into two cases: Either delete an edge or merge its two endpoints.

Exact edge branching algorithm

Edge branching

Branch into two cases: Either delete an edge or merge its two endpoints.

Data reduction (adaption of Edge Cut Rule). Let $V' \subseteq V$ be a "colorful" subgraph. If the edge cut between V' and $V \setminus V'$ is at least as large as the connectivity of V', then merge V' into a single vertex.



Merge-based (non-exact) heuristic

Remark: In preceding work, Corel at al. (Bioinformatics 2010) suggested the following "min-cut greedy heuristic":

- Find two vertices of same color in one connected component.
- Find a minimum edge cut between these two vertices and remove it.
- Repeat this until all connected components are colorful.

Idea for new heuristic

Repeatedly merge the two vertices "most likely" to be in the same component, while immediately deleting edges connecting vertices with intersecting color sets.

Merge-based (non-exact) heuristic

Remark: In preceding work, Corel at al. (Bioinformatics 2010) suggested the following "min-cut greedy heuristic":

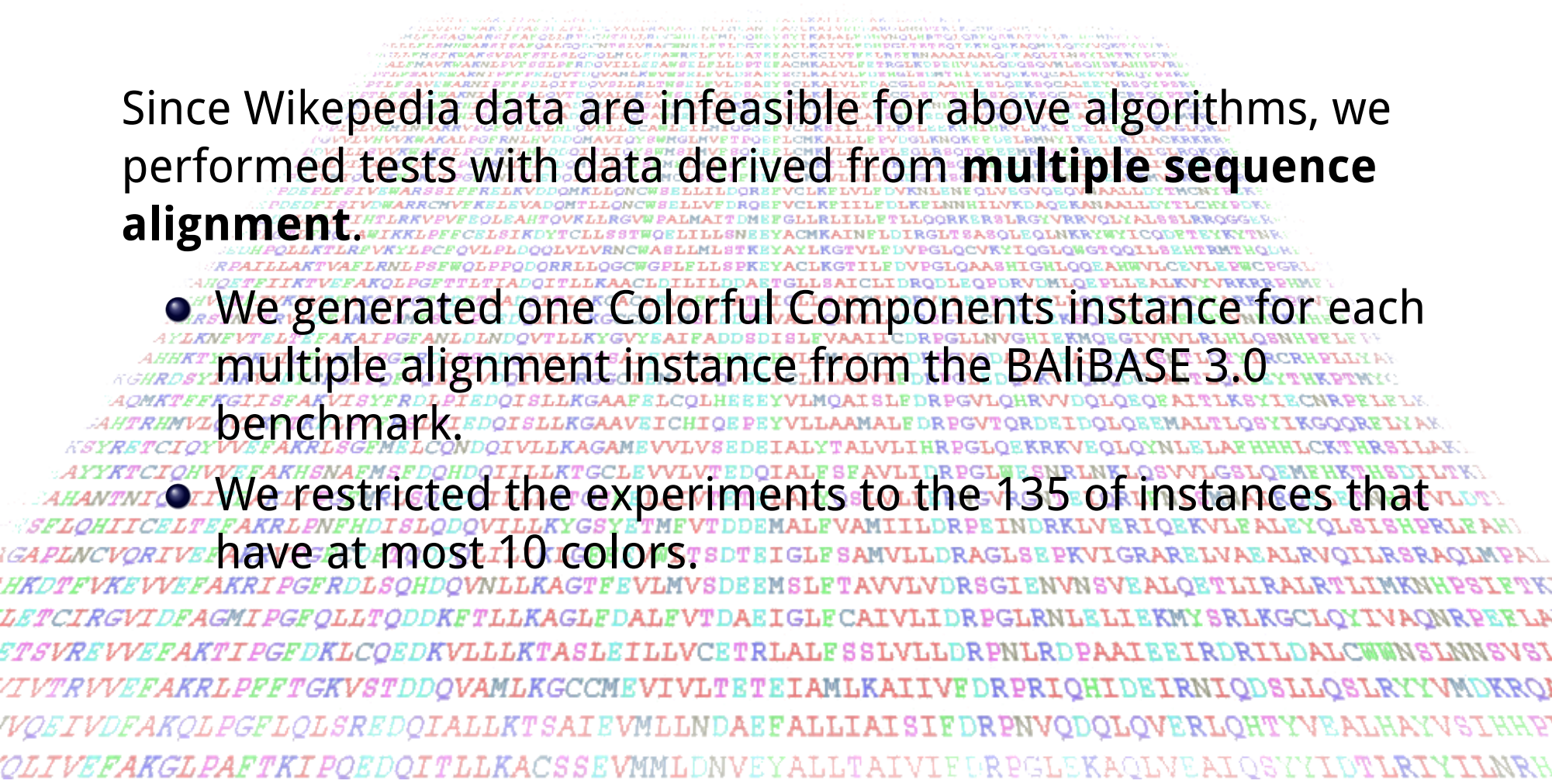
- Find two vertices of same color in one connected component.
- Find a minimum edge cut between these two vertices and remove it.
- Repeat this until all connected components are colorful.

Idea for new heuristic

Repeatedly merge the two vertices "most likely" to be in the same component, while immediately deleting edges connecting vertices with intersecting color sets.

We always merge the endpoints of the edge that maximizes "cut cost" minus "merge cost"...

Real-World Data



Since Wikepedia data are infeasible for above algorithms, we performed tests with data derived from **multiple sequence alignment**.

- We generated one Colorful Components instance for each multiple alignment instance from the BAliBASE 3.0 benchmark.
- We restricted the experiments to the 135 of instances that have at most 10 colors.

Data reduction: Largest connected component

(1) originally

(2) after data reduction without using edge weights etc.

(3) after data reduction using edge weights.

	(1)			(2)			(3)		
	n	m	c	n	m	c	n	m	c
average	504	921	6.2	407	697	4.7	354	607	5.3
median	149	232	6	46	90	5	42	58	5

Branching algorithms and quality of new heuristic

	< 1 s	1 s to 10 min	> 10 min
c^{k-1} search tree	61	6	68
merge branching	70	9	56

Branching algorithms and quality of new heuristic

	< 1 s	1 s to 10 min	> 10 min
c^{k-1} search tree	61	6	68
merge branching	70	9	56

Relative error of non-exact heuristics:

	min.	max.	avg.	med.
min-cut heuristic [1]	0 % (1)	70.0 %	29.2 %	27.8 %
merging heuristic	0 % (76)	12.7 %	0.6 %	0 %

[1] Corel, Pitschi & Morgenstern (Bioinformatics 2010)

Exact Solutions based on Integer Linear Programming

We tried three solving methods using ILP formulations and the CPLEX solver as backend:

- Implicit Hitting Set (Moreno-Centeno/Karp, Manuscript 2011).
- Hitting Set ILP formulation with row generation.
- Clique Partitiong ILP formulation (Régnier 1965; Grötschel and Wakabayashi 1989).

Cutting planes added to the last two methods.

Method 1: Implicit Hitting Set

Hitting Set

Instance: A ground set U and a set of *circuits* $S_1, \ldots, S_n$ with $S_i \subseteq U$ for $1 \leqslant i \leqslant n$.
Task: Find a minimum-size *hitting set*, that is, a set $H \subseteq U$ with $H \cap S_i \neq \emptyset$ for all $1 \leqslant i \leqslant n$.

Method 1: Implicit Hitting Set

Hitting Set

Instance: A ground set U and a set of *circuits* $S_1, \ldots, S_n$ with $S_i \subseteq U$ for $1 \leqslant i \leqslant n$.
Task: Find a minimum-size *hitting set*, that is, a set $H \subseteq U$ with $H \cap S_i \neq \emptyset$ for all $1 \leqslant i \leqslant n$.

Observation

We can reduce Colorful Components to Hitting Set: The ground set U is the set of edges, and the circuits to be hit are the paths between identically-colored vertices.

Method 1: Implicit Hitting Set

Hitting Set

Instance: A ground set U and a set of *circuits* $S_1, \ldots, S_n$ with $S_i \subseteq U$ for $1 \leqslant i \leqslant n$.
Task: Find a minimum-size *hitting set*, that is, a set $H \subseteq U$ with $H \cap S_i \neq \emptyset$ for all $1 \leqslant i \leqslant n$.

Observation

We can reduce Colorful Components to Hitting Set: The ground set U is the set of edges, and the circuits to be hit are the paths between identically-colored vertices.

Problem: Exponentially many circuits!

Method 1: Implicit Hitting Set

How to circumvent the generation of a potentially exponential number of circuits?

Method 1: Implicit Hitting Set

How to circumvent the generation of a potentially exponential number of circuits?

In an **implicit hitting set** problem, the circuits have an implicit description, and a **polynomial-time oracle** (asking the Colorful Components instance) is available that, given a putative hitting set H, either confirms that H is a hitting set or produces a circuit that is not hit by H.

Method 1: Implicit Hitting Set

How to circumvent the generation of a potentially exponential number of circuits?

In an **implicit hitting set** problem, the circuits have an implicit description, and a **polynomial-time oracle** (asking the Colorful Components instance) is available that, given a putative hitting set H, either confirms that H is a hitting set or produces a circuit that is not hit by H.

Several approaches to solving implicit hitting set problems are known, which use an ILP solver as a black box for the Hitting Set subproblems.

Method 2: Hitting Set with row generation

Idea

Instead of using the ILP solver as a black box, we can use *row generation* ("*lazy constraints*") to add constraints inside the solver.

Method 3: Clique Partitioning ILP formulation

Clique Partitioning

Instance: A vertex set V with a weight function $s : \binom{V}{2} \to \mathbb{Q}$.
Task: Find a cluster graph (that is, a disjoint union of cliques) (V, E) that minimizes $\sum_{\{u,v\} \in E} s(u, v)$.

Method 3: Clique Partitioning ILP formulation

Clique Partitioning

Instance: A vertex set V with a weight function $s : \binom{V}{2} \to \mathbb{Q}$.
Task: Find a cluster graph (that is, a disjoint union of cliques) (V, E) that minimizes $\sum_{\{u,v\} \in E} s(u, v)$.

$$s(u,v) = \begin{cases} \infty & \text{if } \chi(u) = \chi(v), \\ -1 & \text{if } \{u,v\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Method 3: Clique Partitioning ILP formulation

Clique Partitioning

Instance: A vertex set V with a weight function $s : \binom{V}{2} \to \mathbb{Q}$.
Task: Find a cluster graph (that is, a disjoint union of cliques) (V, E) that minimizes $\sum_{\{u,v\} \in E} s(u, v)$.

$$s(u,v) = \begin{cases} \infty & \text{if } \chi(u) = \chi(v), \\ -1 & \text{if } \{u,v\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Using bin. vars for edges, add for an induced $P_3 = (u, v, w)$

$$e_{uv} + e_{vw} - e_{uw} \leqslant 1$$
$$e_{uv} - e_{vw} + e_{uw} \leqslant 1$$
$$-e_{uv} + e_{vw} + e_{uw} \leqslant 1$$

Cutting Planes

Definition

A *cutting plane* is a valid constraint that cuts off fractional solutions.

Cutting Planes

Definition

A *cutting plane* is a valid constraint that cuts off fractional solutions.

Tree cut

Let $T = (V_T, E_T)$ be a subgraph of G that is a tree such that all leaves L of the tree have color c, but no inner vertex has. Then

$$\sum_{e \in E_T} d_e \geqslant |L| - 1$$

is a valid inequality, where $d_e = 1$ means "delete edge e".

Cutting Planes

Definition

A *cutting plane* is a valid constraint that cuts off fractional solutions.

Tree cut

Let $T = (V_T, E_T)$ be a subgraph of G that is a tree such that all leaves L of the tree have color c, but no inner vertex has. Then

$$\sum_{e \in E_T} d_e \geqslant |L| - 1$$

is a valid inequality, where $d_e = 1$ means "delete edge e".

We find only tree cuts with 1 or 2 internal vertices.

Experiments with Wikipedia interlanguage links

- 7 languages: English, Spanish, German, French, Russian, Chinese and Hebrew
- 4,090,160 vertices, 9,666,439 edges
- 1,332,253 connected components, of which 1,252,627 are already colorful
- largest connected component has 409 vertices and 747 edges

Experiments with Wikipedia interlanguage links

- 7 languages: English, Spanish, German, French, Russian, Chinese and Hebrew
- 4,090,160 vertices, 9,666,439 edges
- 1,332,253 connected components, of which 1,252,627 are already colorful
- largest connected component has 409 vertices and 747 edges

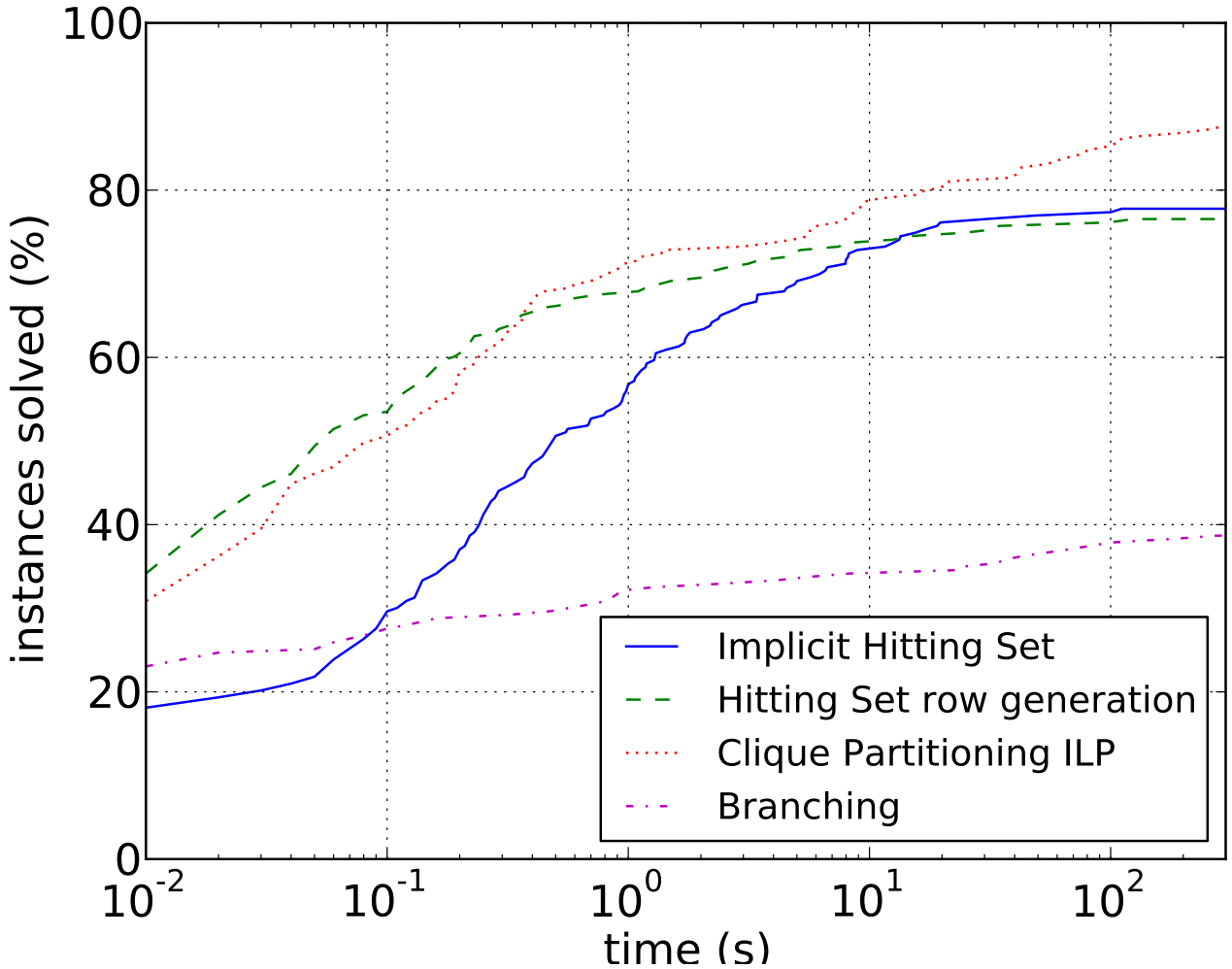
- CLIQUE PARTITIONING algorithm finds solution in 208 s
- Our inexact heuristic finds solution in 46 s (0.21% off)
- Optimal solution deletes 184,759 edges
- 52,058 suggested new links

Random graph model

Model is the recovery of colorful components that have been perturbed.

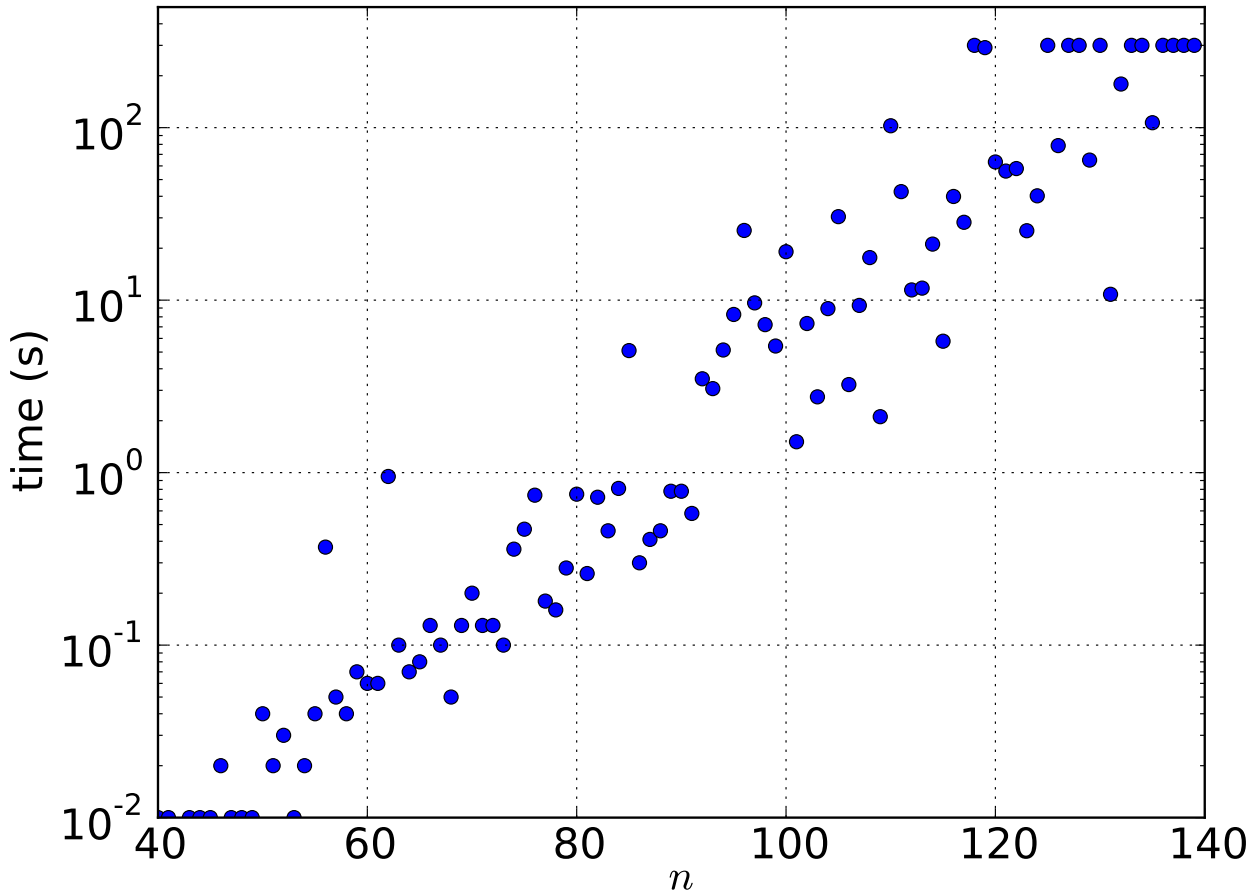
- number of colors: $\{3, 5, 8\}$
- number of vertices: $\{60, 100, 170\}$
- probability that a component contains a vertex of a certain color: $\{0.4, 0.6, 0.9\}$
- probability that between two vertices in a component there is an edge: $\{0.4, 0.6, 0.9\}$
- probability that between two vertices from different components there is an edge: $\{0.01, 0.02, 0.04\}$.

Running times for benchmark set

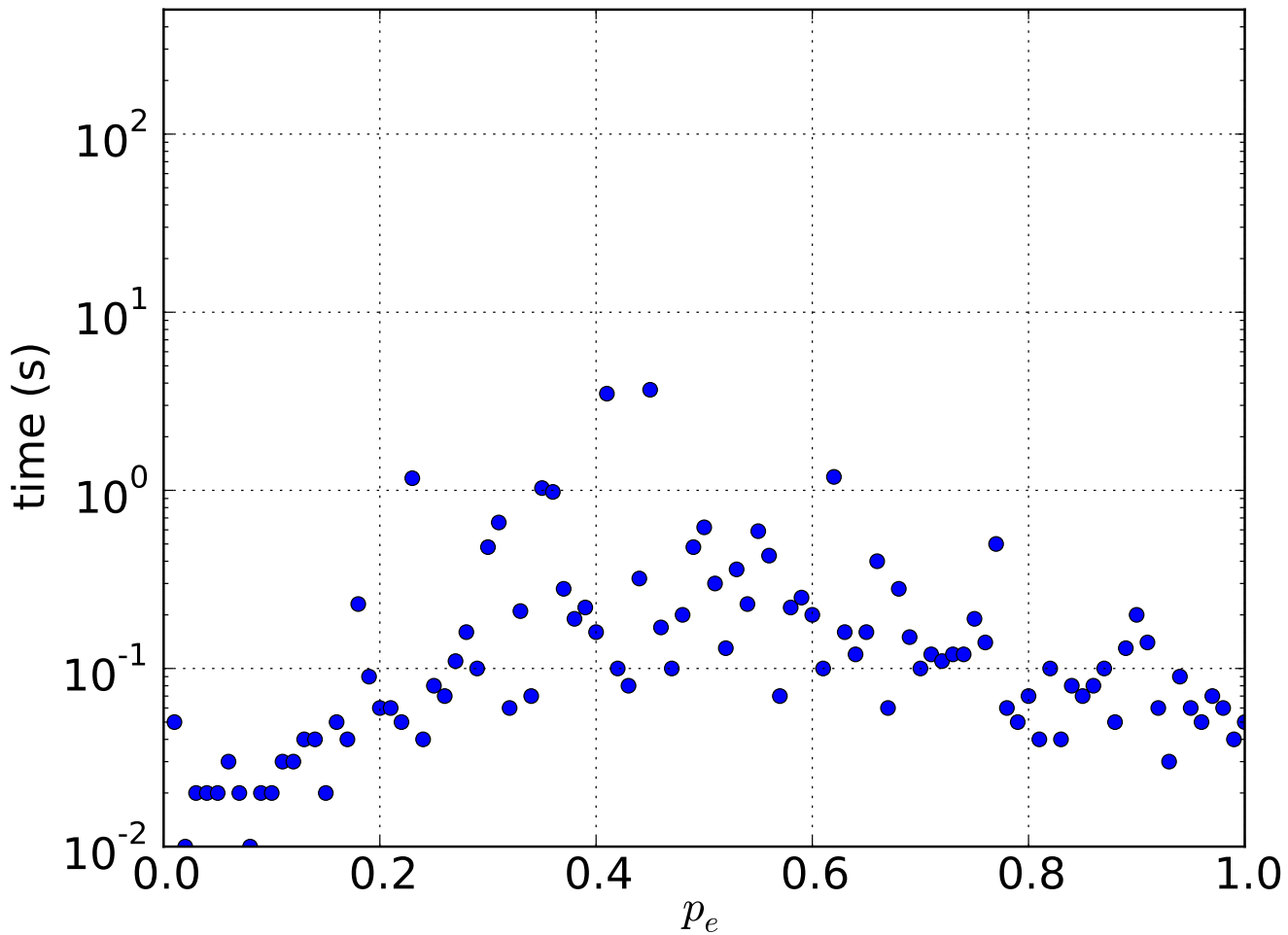


Running time dependence on number of vertices

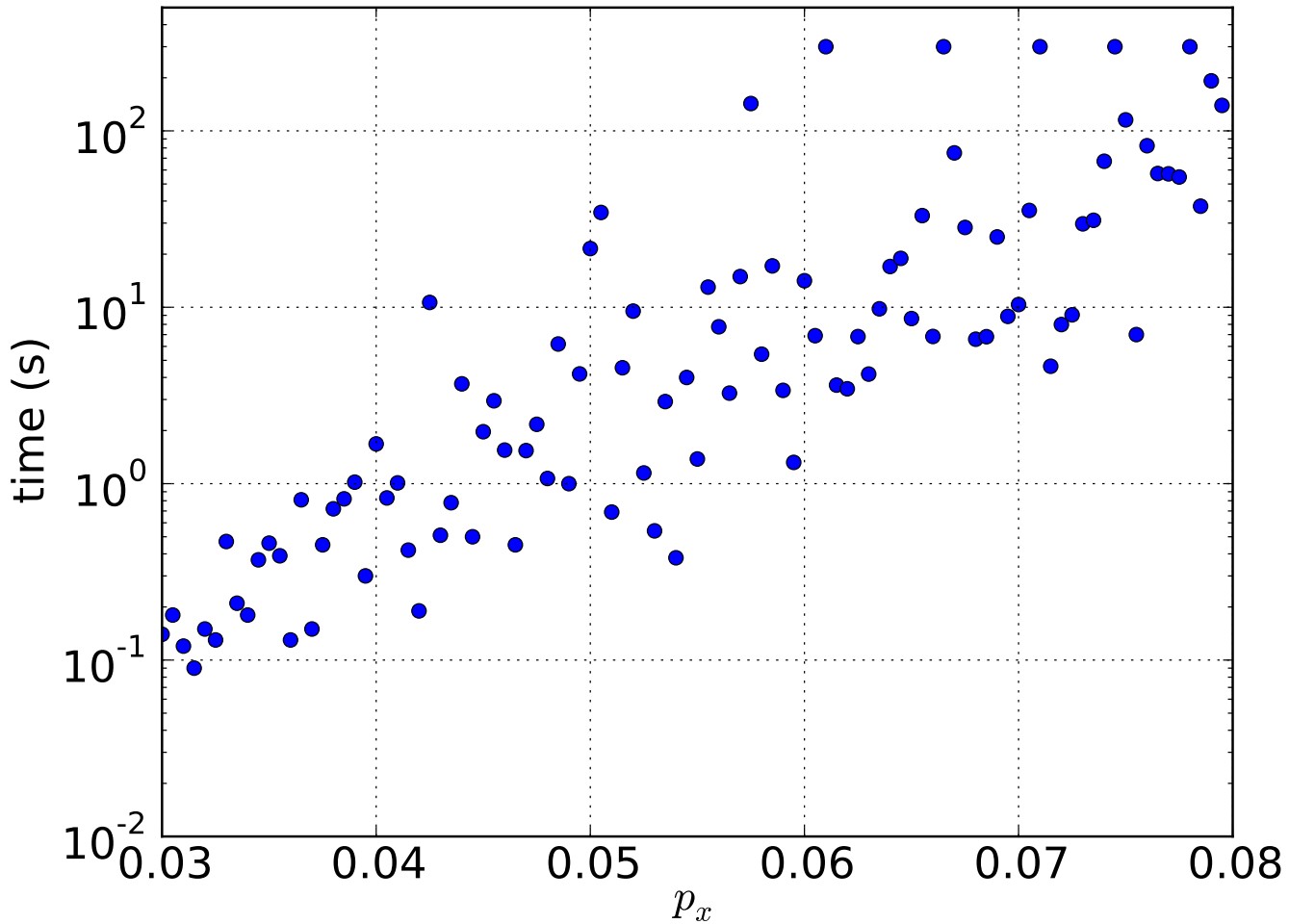
Here and in the following: Base parameter values: $n = 70$, $c = 6$, $p_v = 0.6$, $p_e = 0.7$, $p_x = 0.03$.



Running time dependence on probability of intracluster edges



Running time dependence on probability of intercluster edges



General Conclusions and Outlook

- ILP-based solutions clearly outperform combinatorial algorithms.

General Conclusions and Outlook

- ILP-based solutions clearly outperform combinatorial algorithms.
- Clique Partioning method currently fastest.

General Conclusions and Outlook

- ILP-based solutions clearly outperform combinatorial algorithms.
- Clique Partioning method currently fastest.
- Implicit Hitting Set Method less tied to a specific ILP solver than Clique Partitioning...

General Conclusions and Outlook

- ILP-based solutions clearly outperform combinatorial algorithms.
- Clique Partioning method currently fastest.
- Implicit Hitting Set Method less tied to a specific ILP solver than Clique Partitioning...
- Still beneficial: data reduction before starting solvers.

General Conclusions and Outlook

- ILP-based solutions clearly outperform combinatorial algorithms.
- Clique Partioning method currently fastest.
- Implicit Hitting Set Method less tied to a specific ILP solver than Clique Partitioning...
- Still beneficial: data reduction before starting solvers.
- Exact methods help spotting the qualities of a new inexact heuristic which may be used for hard instances.

General Conclusions and Outlook

- ILP-based solutions clearly outperform combinatorial algorithms.
- Clique Partioning method currently fastest.
- Implicit Hitting Set Method less tied to a specific ILP solver than Clique Partitioning...
- Still beneficial: data reduction before starting solvers.
- Exact methods help spotting the qualities of a new inexact heuristic which may be used for hard instances.
- Relaxation of the colorfulness constraint (more than one color occurrence per component) might be useful for further applications.

General Conclusions and Outlook

- ILP-based solutions clearly outperform combinatorial algorithms.
- Clique Partioning method currently fastest.
- Implicit Hitting Set Method less tied to a specific ILP solver than Clique Partitioning...
- Still beneficial: data reduction before starting solvers.
- Exact methods help spotting the qualities of a new inexact heuristic which may be used for hard instances.
- Relaxation of the colorfulness constraint (more than one color occurrence per component) might be useful for further applications.
- Open: useful applications in network alignment.

General Conclusions and Outlook

- ILP-based solutions clearly outperform combinatorial algorithms.
- Clique Partioning method currently fastest.
- Implicit Hitting Set Method less tied to a specific ILP solver than Clique Partitioning...
- Still beneficial: data reduction before starting solvers.
- Exact methods help spotting the qualities of a new inexact heuristic which may be used for hard instances.
- Relaxation of the colorfulness constraint (more than one color occurrence per component) might be useful for further applications.
- Open: useful applications in network alignment.
- Open to compare with practical solvability of more general problems such as MultiCut, Mutli-Multiway Cut, and Clique Partitioning.

Literature References

Talk based on:

- Sharon Bruckner, Falk Hüffner, Christian Komusiewicz, Rolf Niedermeier, Sven Thiel, Johannes Uhlmann: Partitioning into Colorful Components by Minimum Edge Deletions. Proc. CPM 2012: 56-69.
- Sharon Bruckner, Falk Hüffner, Christian Komusiewicz, Rolf Niedermeier: Entity Disambiguation by Partitioning under Heterogeneity Constraints. Manuscript, June 2012.

Thank you!