Mathematisches Forschungsinstitut Oberwolfach

Report No. 7/2006

Algorithmic Graph Theory

Organised by
Artur Czumaj (New Jersey)
Friedhelm Meyer auf der Heide (Paderborn)
Klaus Jansen (Kiel)
Ingo Schiermeyer (Freiberg)

February 12th – February 18th, 2006

Abstract. The main focus of this workshop was on mathematical techniques needed for the development of efficient solutions and algorithms for computationally difficult graph problems. The techniques studied at the workshhop included: the probabilistic method and randomized algorithms, approximation and optimization, structured families of graphs and approximation algorithms for large problems. The workshop Algorithmic Graph Theory was attended by 46 participants, many of them being young researchers. In 15 survey talks an overview of recent developments in Algorithmic Graph Theory was given. These talks were supplemented by 10 shorter talks and by two special sessions.

Mathematics Subject Classification (2000): 05Cxx.

Introduction by the Organisers

The workshop *Algorithmic Graph Theory*, organized by Artur Czumaj (New Jersey), Friedhelm Meyer auf der Heide (Paderborn), Klaus Jansen (Kiel) and Ingo Schiermeyer (Freiberg) was held February 12th–February 18th, 2006. This meeting was attended by 46 participants from a wide range of geographic regions, many of them young researchers. In the morning sessions survey talks providing an overview of recent developments in Algorithmic Graph Theory were presented.

- Geometry and Graphs
  - Random triangulations of planar point sets (Emo Welzl)
  - Dynamic routing in graphs with applications to harbour logistics (Rolf Möhring)
  - Page migration in dynamic networks (Friedhelm Meyer auf der Heide)
  - Simple coresets for clustering problems (Christian Sohler)

- Graph Algorithms
  - Parallel matching algorithms (Stefan Hougardy)
  - Scheduling malleable tasks with precedence constraints (Klaus Jansen)
  - Coloring random graphs (Lefteris Kirousis)
  - Faster approximation algorithms for packing and covering problems (Daniel Bienstock)
  - Algebraic graph algorithms (Piotr Sankowski)
- Game Theory and Graphs
  - Graphs, Games and Algorithms (Paul Spirakis)
  - Learning wardrop equilibria (Bertold Vöcking)
- Graph Structures
  - Phylogenetic Trees and k-leaf powers (Andreas Brandstädt)
  - Arbitrarily vertex decomposable graphs (Mariusz Woźniak)
  - Precoloring extension (Margit Voigt)
  - On exact algorithms for treewidth (Hans Bodlaender)

There were 10 shorter talks and two special sessions on "Graph Algorithms" (organized by Christian Sohler) and "Graph Colouring" (organized by Ingo Schiermeyer). The contributions showed progress in the field provided in recent years. Furthermore, several open problems and conjectures were presented, some of them still far from being resolved. Beyond the program there was plenty of time for the participants to use the stimulating atmosphere of the Oberwolfach Institute for fruitful discussions.

Workshop: Algorithmic Graph Theory

Table of Contents

Abstracts

Upper bounds for the chromatic number of a graph

Ingo Schiermeyer
(joint work with Bert Randerath)

1. On Reed's conjecture about ω, Δ and χ

For a given graph G, the clique number $\omega(G)$, the chromatic number $\chi(G)$ and the maximum degree $\Delta(G)$ satisfy $\omega(G) \leq \chi(G) \leq \Delta(G) + 1$. In 1941 Brooks [1] has shown that complete graphs and odd cycles are the only graphs attaining the upper bound $\Delta(G) + 1$.

Theorem 1. *If a connected graph $G = (V, E)$ is neither complete nor an odd cycle, then G has a $\Delta(G)$-colouring.*

In 1998 Reed [3] posed the following conjecture:

Conjecture 1. *For any graph G of maximum degree Δ,*

$$\chi(G) \leq \lceil \frac{\Delta + 1 + \omega}{2} \rceil.$$

The Chvátal graph, the smallest 4-regular, triangle-free graph of order 12 with chromatic number 4, shows that the rounding up in this conjecture is necessary. In this talk we will present some old and new partial solutions for this conjecture. In particular we will show that the conjecture is true

(1) for all graphs of order $n \leq 12$,
(2) for all graphs with $\Delta(G) = n - k$ and $\alpha(G) \geq k$ for fixed k and
(3) for all graphs with $n - 5 \leq \Delta(G) \leq n - 1$.

2. A new upper bound for the chromatic number of a graph

For a connected graph G of order n, the clique number $\omega(G)$, the chromatic number $\chi(G)$ and the independence number $\alpha(G)$ satisfy $\omega(G) \leq \chi(G) \leq n - \alpha(G) + 1$. We will show that
$\chi(G) \leq \frac{n+\omega+1-\alpha}{2}$, which is the arithmetic mean of the previous lower and upper bound.

Theorem 2. [4] *Let G be a connected graph of order n with clique number $\omega(G)$, chromatic number $\chi(G)$ and independence number $\alpha(G)$. Then $\chi(G) \leq \frac{n+\omega+1-\alpha}{2}$. Moreover, if G contains no $(K_{\omega+3} - C_5)$ and is not a split graph, then $\chi(G) \leq \frac{n+\omega-\alpha}{2}$.*

Combining Theorem 1 and Theorem 2 we obtain the following improved upper bound for the chromatic number of a graph.

Theorem 3. [4] *Let G be a connected graph of order n with clique number $\omega(G)$, chromatic number $\chi(G)$, maximum degree $\Delta(G)$ and independence number $\alpha(G)$. Then $\chi(G) \leq min\{\Delta(G)+1, \frac{n+\omega+1-\alpha}{2}\}$. Moreover, if G contains no $(K_{\omega+3} - C_5)$ and is neither a split graph nor an odd cycle, then $\chi(G) \leq min\{\Delta(G), \frac{n+\omega-\alpha}{2}\}$.*

References

[1] R. L. Brooks, *On colouring the nodes of a network*, Proc. Cambridge Phil. Soc. **37** (1941) 194-197.

[2] B. Randerath and I. Schiermeyer, *On Reed's Conjecture about ω, Δ and χ,* Preprint 2004.

[3] B. A. Reed, ω, Δ *and* χ, J. Graph Theory **27 (4)**, (1998), 177-212.

[4] I. Schiermeyer, *A new upper bound for the chromatic number of a graph*, Preprint 2006.

Chromatic Polynomials: Generalizations and Algorithmic Aspects

Peter Tittmann

The chromatic polynomial $P(G,x)$ of a finite undirected graph $G = (V,E)$ gives the number of proper vertex colorings of G with x colors. In [1] we introduced a two-variable polynomial which simultaneously generalizes the chromatic polynomial, the independence polynomial, and the matching polynomial of a graph. The new polynomial is defined by

$$P(G;x,y) = \sum_{X \subseteq V} (x-y)^{|X|} P(G-X;y).$$

It may also be considered as an evaluation of Stanley's *chromatic symmetric function* X_G [3].

It is known [4] that the computation of the chromatic polynomial is a #P-hard problem. In [1], we showed that the computation of the generalized chromatic polynomial $P(G;x,y)$ is in **P** for graphs of bounded treewidth and for graphs with a complement of bounded treewidth. This result implies that the computation of the chromatic polynomial, the matching polynomial, and the independence polynomial is a polynomial problem within these graph classes.

A partition π of the vertex set V is a *clique partition* if each block of π induces a clique in G. Let C_G be the set of all clique partitions of G. We denote by c_i the number of partitions of C_G with exactly i blocks. The *clique partition polynomial* of G is defined by

$$C(G,x) = \sum_{\pi \in C_G} x^{|\pi|} = \sum_{i=0}^{n} c_i x^i.$$

This polynomial is closely related to the *adjoint polynomial* [2].

Let U be a separating vertex set of G such that the *split components* with respect to U are subgraphs G^1 and G^2 of G. Both polynomials, $C(G,x)$ and $P(G,x)$, satisfy a *splitting formula*, that is a representation as a sum with respect to split components:

$$f(G,x) = \sum_{\sigma,\tau} a(\sigma,\tau)\, f\left(G^1_\sigma, x\right) f\left(G^2_\tau, x\right)$$

The sum runs over all partitions σ and τ of U. The polynomial f stands for either C or P. The function a depends only on σ and τ. The splitting formula leads to efficient algorithms for the computation of $f(G,x)$ in graphs of bounded treewidth. There is, in addition, a simple way to obtain the coefficients of $P(G,x)$ from $C\left(\bar{G},x\right)$ and the coefficients of $C(G,x)$ from $P\left(\bar{G},x\right)$. Thus we can switch arbitrarily often to the complement while computing the chromatic polynomial of a graph. In this way we obtain a graph class permitting an efficient computation of the chromatic polynomial that generalizes both the class of graphs with bounded treewidth and those graphs having a complement of bounded treewidth. The open problem is the characterization of this class by simple properties, forbidden minors, etc.

References

[1] Klaus Dohmen, Andre Pönitz, Peter Tittmann: *A new two-variable generalization of the chromatic polynomial,* Discrete Mathematics & Theoretical Computer Science, Volume 6, n° 1 (2003), 69-90.

[2] Ruying Liu, Haixing Zhao, Xueliang Li, Ligong Wang: *On properties of adjoint polynomials of graphs and their applications*, Australas. J. Combin. 30 (2004) 291-306.

[3] Richard P. Stanley: *Enumerative combinatorics, Volume II*, Cambridge University Press, 1999.

[4] D.J.A. Welsh: *Knots,colourings and counting*, Cambridge University Press, Cambridge, 1993.

Coloring Random Graphs
A Short Survey

Lefteris M. Kirousis

RA Computer Technology Institute
and
Department of Computer Engineering and Informatics, University of Patras

The model. Unless otherwise stated, we consider random graphs in the classical Erdős-Rényi models $G_{n,p}$ or $G_{n,m}$. Although in some respects these two models are not equivalent between them, for the purposes of this short survey we use them interchangeably. We consider sparse graphs, i.e. graphs where $p = d/n$ or alternatively $m = dn/2$. Average number of edges in $G(n, p = d/n)$ is $\sim dn/2$ and therefore average degree is d. The value $d/2$ is known as the *edge-density*.

Phase transitions: non-rigorous results. It was experimentally verified [1] that or each fixed k (amenable to experimentation), there is a *threshold average degree* d^*_k such that

- If $d < d_k^*$, then a random graph with average degree d is a.a.s. k-colorable, while
- if $d > d_k^*$ then such a graph is a.a.s. non-k-colorable.

NB: "a.a.s." means with probability approaching 1 asymptotically with the number of vertices.
Analytic (but non-rigorous) verification of the previous experimental results by the cavity and replica symmetry breaking methods of Statistical Physics.
For $k = 3$ experiments and analytic non-rigorous results suggest that $d_3^* \simeq 4.69$.

The Achlioptas–Friedgut theorem [2]. *For all $k \geq 3$, there is a sequence $d_k^*(n)$ such that $\forall\epsilon$, a random graph with average degree $d_k^*(n) - \epsilon$ is a.a.s. k-colorable and a random graph with average degree $d_k^*(n) + \epsilon$ is a.a.s. non-k-colorable.*
Still open question for $k \geq 3$: Does $d_k^*(n)$ converge? If yes, to what value?

Corollary 1. *Given d, if the probability that a random graph with average degree d is k-colorable is bounded away from zero, then for any $d' < d$, a random graph with average degree d' is a.a.s. k-colorable.*

Upper bounds. The $G(n, m)$ model is better suited in this case. Reminder: experimentally, the putative threshold of 3-colorability occurs for edge-density 2.35 (average degree 4.69).

- First upper bound for 3-colorability: 2.71 — observed by several researchers independently – *Markov's inequality.*
- Current best: 2.427 [3] – *typical graphs plus the decimation technique.*

Basic technique: Let G be a random graph and $C(G)$ the random class of its 3-colorings. Then

$$\Pr[G \text{ is 3-colorable}] = \Pr[|C(G)| \geq 1] \leq \mathbb{E}(|C(G)|).$$

$\mathbb{E}(|C(G)|)$ is easy to compute. So we find the values of edge-density for which it vanishes and thus we get a trivial upper bound, namely 2.71. This method is improved by two techniques:

- Make $C(G)$ (the set of all 3-colorings) "thinner" by considering *rigid* 3-colorings, i.e. colorings where any change of color to a higher one (in the RGB ordering) destroys the legality [4, 5].
- Examine not the whole space of possible graphs, but a subset of it comprised of graphs that are (i) are *typical* with respect to their degree sequence (Poisson) and (ii) have been repetitively depleted of vertices of degree 2 or less, as these vertices do not interfere with the colorability (*the decimation technique* [3]). This gives the current best upper bound.

Lower bounds. To show that d_k^- is a lower bound for d_k^*, consider a simple algorithm for k-coloring (like list coloring with simple heuristics for the selection of the next vertex to be colored, and without backtracking) and prove that for $d < d_k^-$, the coloring algorithm a.a.s. succeeds. The best lower bound for 3-Col obtained in this way: 4.03 [6]. There is evidence that the algorithmic approach cannot overcome a barrier smaller than the value of the experimental threshold.

Also, this technique gives no improvement over the previously known best lower bound of $2k \ln k - o(k \ln k)$ for d_k^*, for arbitrary k. NB: By the first moment method: $d_k^* < 2k \ln k - \ln k$. Also known that the chromatic number of graphs in $G(n, p = d/n)$ ranges a.a.s. within a window of only *two* possible integer values. However, no information on these two values. Task: Make the known asymptotic of the lower bound of d_k^* finer so that the two possible values of the chromatic number are found.

The second moment method. Let X be a non-negative variable (usually a *counting* variable) that depends on n. Then:

$$\Pr[X > 0] \geq \frac{(\mathbb{E}(X))^2}{\mathbb{E}(X^2)}.$$

Achlioptas and Naor [7]. Let k_d be the smallest integer k such that $d < 2k \ln k$. Almost all $G_{n,p=d/n}$ random graphs have chromatic number either k_d or $k_d + 1$. Method of Proof: Second moment where X counts the number of balanced k-colorings of $G_{n,p=d/n}$. Balanced: each color is assigned to an equal number of vertices.

Random regular graphs. Progress is slower. Was known that the chromatic number of random regular graphs of degree d a.a.s. ranges in $\{k_d, k_d + 1, k_d + 2\}$, where k_d is the smallest integer k such that $d < 2k \ln k$. Also, known that almost all 4-regular graphs have chromatic number 3, and that almost all 6-regular graphs have chromatic number 4. Moreover, non-rigorous results from physics indicate that almost all 5-regular graphs have chromatic number 3 but that it would be difficult to prove this with local search algorithmic techniques. Until recently, the only rigorous result for 5-regular graphs was that almost all of them have chromatic number 3 or 4. Finally, second moment fails when X counts 3-colorings, even if they are balanced.
Indeed, by linearity of expectation and by summing over pairs of 3-colorings we have:

$$\mathbb{E}(X^2) = \sum_i E_i P_i,$$

where E_i is the number of pairs of 3-colorings with a given *pattern* of color assignments (characterized by a parameter i) (entropy factor) and P_i is the probability that a fixed pair of color assignments with pattern E_i is legal (energy factor).
Now, the term $E_i P_i$ that is equal (ignoring sub-exponential factors) to $(\mathbb{E}(X))^2$ is the *barycentric* term that corresponds to a completely symmetric pattern, but unfortunately unlike the case of $G(n, p)$ graphs, the barycentric term is not the prevalent one in the sum. It turns out that an exponentially larger than the barycentric term is obtained for pairs of colorings that are slightly biased towards assigning the same color to a vertex.
To eliminate this bias, the authors in [8] considered colorings where each vertex has neighbors with both the other two legal colors. Thus they proved that a 5-regular graph is 3-colorable with positive probability independent of its size. Recently Kemkes and Wormald [9] raised this probability to 1 (asymptotically).

References

[1] P. Cheeseman, R. Kanefsky, and W. Taylor, *Where the really hard problems are*, IJCAI (1991), 163–169.

[2] D. Achlioptas and E. Friedgut, *A sharp threshold for k-colorability*, Random Struct. Algorithms **14**(1) (1999), 63–70.

[3] O. Dubois and J. Mandler, Unpublished result.

[4] D. Achlioptas and M. Molloy, *Almost all graphs with* $2.522n$ *edges are not* 3*-colorable*, Electr. J. Comb. **6** (1999).

[5] A.C. Kaporis, L.M. Kirousis, and Y.C. Stamatiou, *A Note on the non-colorability threshold of a random graph*, Electr. J. Comb. **7** (2000).

[6] D. Achlioptas and C. Moore, *Almost all graphs with average degree* 4 *are* 3*-colorable*, J. Comput. Syst. Sci. **67**(2) (2003), 441–471.

[7] D. Achlioptas and A. Naor, *The two possible values of the chromatic number of a random graph*, STOC (2004), 587–593.

[8] G. Grammatikopoulos, J. Díaz, A.C. Kaporis, L.M. Kirousis, X. Pérez, and D.G. Sotiropoulos, 5-*Regular Graphs are* 3-*Colorable with Positive Probability*, ESA (2005), 215–225.

[9] G. Kemkes and N. Wormald, *On the chromatic number of a 5-regular random graph*, Research Report CORR 2005-22, Faculty of Mathematics, University of Waterloo (2005).

Large Induced Forests in C_3- and C_4-free Graphs of Maximum Degree at most 3

Dieter Rautenbach

The Four Color Theorem immediately implies that every planar graph (*all considered graphs are finite, simple and undirected*) of order n has an independent set of order at least $\frac{n}{4}$. In view of the complicated proof of the Four Color Theorem there is interest in simpler proofs of this and similar statements and several related conjectures have been proposed.

Albertson and Berman [2] conjectured that every planar graph of order n has an induced forest of order at least $\frac{n}{2}$ and Akiyama and Watanabe [1] conjectured that every bipartite planar graph of order n has an induced forest of order at least $\frac{5n}{8}$. Borodin [5] proved that every planar graph of order n has an induced forest of order at least $\frac{2n}{5}$ and Borodin and Glebov [6] proved that every C_3- and C_4-free planar graph can be partitioned into an independent set and an induced forest.

Motivated by this kind of problem, Alon, Mubayi and Thomas [4] study the existence of large induced forests in K_4-free graphs of maximum degree at most 3. As an essential tool for their result they prove (cf. Lemma 2.1 in [4]) that every C_3-free graph $G = (V, E)$ of order n and size m has an induced forest of order at least

$$n - \frac{m}{4} = \sum_{u \in V} \left(1 - \frac{d_G(u)}{8}\right). \tag{1}$$

Here we consider the closely related problem of the existence of large induced forests in C_3- and C_4-free graphs of maximum degree at most 3 and present a best-possible result.

In [3] Alon, Kahn and Seymour [3] studied the existence of large induced d-degenerate subgraphs (*a graph is d-degenerate if every induced subgraph has a vertex of degree less than d. Note that independent sets and forests correspond exactly to 1-degenerate and 2-degenerate graphs, respectively.*). Generalizing the lower bound on the independence number due to Caro [7] and Wei [9] they prove that every graph $G = (V, E)$ has an induced d-degenerate subgraph of order

$$\sum_{u \in V} \min\left\{1, \frac{d}{d_G(u)+1}\right\} \tag{2}$$

and explicitely ask for an improvement of this estimate for C_3-free graphs. In view of this question we would like to mention the simple observation that the proof given by Shearer [8] for his lower bound (Theorem 1 in [8]) for the independence number in C_3-free graphs immediately implies the following.

Proposition 1. *Every C_3-free graph $G = (V, E)$ has an induced d-degenerate subgraph of order at least $\sum\limits_{u \in V} f_d(d_G(u))$ with $f_d(i) = 1$ for $0 \le i \le d-1$ and $f_d(i) = \frac{1+\left(i^2-i\right)f_d(i-1)}{i^2+1}$ for $i \ge d$.*

Sketch of the proof: Clearly, we can assume that G has no vertex of degree less than d. If we can find a vertex u such that $0 \le 1 - f_d(d_G(u)) - \sum_{v \in N_G(u)} f_d(d_G(u)) + \sum\limits_{w:\text{dist}_G(u,w)=2} (f_d(d_G(w) - |N_G(u) \cap N_G(w)|) - f_d(d_G(w)))$, then adding u to a sufficiently large induced d-degenerate subgraph of $G[V \setminus (\{u\} \cup N_G(u))]$ implies the desired result by an inductive argument. The existence of such a vertex is established by proving - essentially using C_3-freeness and properties of f_d - that the sum of the above terms over all vertices $u \in V$ is non-negative.

For a graph G let $a(G)$ denote the maximum order of an induced forest. We present a lower bound on $a(G)$ for C_3- and C_4-free graphs of maximum degree at most 3. The graph whose components are as the graph in Figure 1 clearly shows that our bound is best-possible. (*The vertices that induce a forest of maximum order are encircled.*)

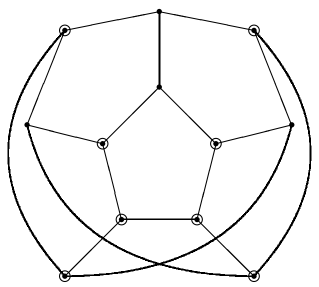


Figure 1: A cubic C_3- and C_4-free graph G with $a(G) = 8 = 12 - 18 \cdot \frac{2}{9} = n(G) - \frac{2}{9}m(G)$.

Theorem 2. *If G is a C_3- and C_4-free graph of maximum degree at most 3, then*

$$a(G) \geq n(G) - \frac{2}{9}m(G).$$

Sketch of the proof: For the proof we choose a counterexample of minimum order and deduce a contradiction via the analysis of quite a long list of local configurations. In order to convey a flavour of the argument we give just one example.
Assume that G contains the graph in Figure 2 as an induced subgraph such that deleting the vertices in $\{x_i \mid 0 \leq i \leq 4\}$ and adding the dotted edge y_0y_4 does not create a cycle of length less than 5, i.e. $G' = G - \{x_i \mid 0 \leq i \leq 4\} + \{y_0y_4\}$ is C_3- and C_4-free.
Since G' is no counterexample, there is a set F' with $|F'| \geq (n-5) - \frac{2}{9}(m-9)$ such that $G'[F']$ is a forest. Since $G[F' \cup \{x_0, x_2, x_4\}]$ is a forest (cf. Figure 2) of order at least $n - \frac{2}{9}m - 5 + 9 \cdot \frac{2}{9} + 3 = n - \frac{2}{9}m$, we obtain a contradiction. □

We close with some open problems.
Whereas (1) is best-possible for cubic graphs there seems to be room for improvement for 4-regular and 5-regular graphs. Examples in [4] suggest that the best-possible estimates could be $a(G) \geq \frac{4}{7}n$ for 4-regular graphs G of order n and $a(G) \geq \frac{1}{2}n$ for 5-regular graphs G of order n.
What happens with the bound in Theorem 2 if we consider larger girths or do not restrict the maximum degree? What are the extremal graphs?
Note that the bounds (1), (2) and also the bounds given in Proposition 1 and Theorem 2 are of a similar form in the sense that every vertex contributes an amount depending on its degree. In view of this, one could study conditions on functions $g : \mathbb{N}_0 = \{0, 1, 2, ...\} \to \mathbb{R}$ with the property that every (C_3-free) graph $G = (V, E)$ has an induced forest (induced d-degenerate subgraph) of order at least $\sum_{u \in V} g(d_G(u))$.

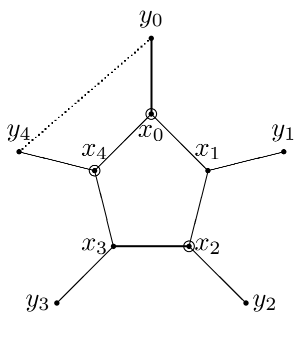


Figure 2: $G' = G - \{x_i \mid 0 \leq i \leq 4\} + \{y_0y_4\}$.

References

[1] J. Akiyama and M. Watanabe, Maximum induced forests of planar graphs, *Graphs Comb.* **3** (1987), 201-202.

[2] M.O. Albertson and D.M. Berman, A conjecture on planar graphs, in: Graph Theory and Related Topics, J. A. Bondy and U. S. R. Murty, (eds.), Academic Press, 1979, 357.

[3] N. Alon, J. Kahn and P.D. Seymour, Large induced degenerate subgraphs, *Graphs Comb.* **3** (1987), 203-211.

[4] N. Alon, D. Mubayi and R. Thomas, Large induced forests in sparse graphs, *J. Graph Theory* **38** (2001), 113-123.

[5] O.V. Borodin, A proof of Grünbaum's conjecture on the acyclic 5-colorability of planar graphs, *Sov. Math., Dokl.* **17** (1976), 1499-1502.

[6] O.V. Borodin and A.N. Glebov, On the partition of a planar graph of girth 5 into an empty and an acyclic subgraph, *Diskretn. Anal. Issled. Oper., Ser. 1* **8** (2001), 34-53.

[7] Y. Caro, New results on the independence number, Technical report, Tel-Aviv University, 1979.

[8] J.B. Shearer, A note on the independence number of triangle-free graphs, II, *J. Comb. Theory, Ser. B* **53** (1991), 300-307.

[9] V.K. Wei, A lower bound on the stability number of a simple graph, Bell Laboratories Technical Memorandum 81-11217-9, 1981.

Exact exponential-time algorithms for NP-hard graph problems: Domination and Independence

Dieter Kratsch

(joint work with Fedor V. Fomin, Fabrizio Grandoni)

The interest in exact and fast exponential time algorithms solving hard problems dates back to the sixties and seventies [4, 5]. The last decade has seen an increasing interest in research in fast exponential-time algorithms. Examples of recently developed exponential-time algorithms for NP-hard graph problems are algorithms for Coloring [1, 2] Max-Cut [6] and Treewidth [3].
We refer to the nice survey written by Woeginger. He emphasizes the main techniques to construct fast exponential time algorithms for NP-hard problems, among them dynamic programming and "branch & reduce" (also called backtracking, and "cutting the search tree"). Branch & reduce seems to be the most promising approach for a variety of NP-hard problems, among them various graph problems.
The analysis of such recursive algorithms is based on the bounded search tree technique: a measure of the size of the subproblems is defined; this measure is used to lower bound the progress made by the algorithm at each branching step. For the last 30 years the research on branch & reduce algorithms has been mainly focused on the design of more and more sophisticated algorithms. However, measures used in the analysis are usually very simple.

In our research we have shown the power of analysing the worst case running time of simple branch & reduce algorithms using a sophisticated measure. This approach has been called "Measure & Conquer" and while it seems useful for the analysis of most branch & reduce algorithms, the following two results of our work seem to be particularly striking.

(1) We obtained the currently fastest exponential time algorithm to compute a minimum dominating set of a graph. Our simple branch & reduce algorithm has running time $O(2^{0.598\,n})$. (For a long time nothing better than the trivial $O(2^n)$ time algorithm had been known.)
(2) We provide a simple branch & reduce algorithm to solve the well-studied maximum independent set problem. Its analysis with a sophisticated measure shows that the running time of the algorithm is $O(2^{0.288\,n})$, which is competitive with the current best time bounds obtained with far more complicated algorithms (and naive analysis).

Our work suggests possible ways for a better analysis of branch & reduce algorithms based on the right choice of the measure. It also supports the intuition that the running time of many branch & reduce exponential time algorithms is overestimated.
This motivated research on lower bounds of the worst case running time for a particular branch & reduce algorithm. We obtained a lower bound of $\Omega(2^{0.333n})$ for our dominating set algorithm and a lower bound of $\Omega(2^{0.167\,n})$ for our independent set algorithm.
The large gap between the upper and the lower bounds of the two algorithms may suggest that the upper bounds are far from being tight, i.e. the worst case running time of our algorithms might be (much) better than the upper bounds that we are able to prove.

References

[1] R. Beigel and D. Eppstein. 3-coloring in time $O(1.3289^n)$. *Journal of Algorithms* 54:168–204, 2005.

[2] J. M. Byskov, *Enumerating maximal independent sets with applications to graph colouring*, Operations Research Letters **32** (2004) 547–556.

[3] F. V. Fomin, D. Kratsch, and I. Todinca, *Exact algorithms for treewidth and minimum fill-in*, Proceedings of the *31st International Colloquium on Automata, Languages and Programming (ICALP 2004)*, Springer, LNCS vol. 3142, 2004, pp. 568–580.

[4] M. Held and R.M. Karp, *A dynamic programming approach to sequencing problems*, Journal of SIAM **10** (1962), 196–210.

[5] R. Tarjan and A. Trojanowski, *Finding a maximum independent set*, SIAM Journal on Computing **6** (1977) 537–546.

[6] R. Williams, *A new algorithm for optimal constraint satisfaction and its implications*, Proceedings of the *31st International Colloquium on Automata, Languages and Programming (ICALP 2004)*, Springer, LNCS vol. 3142, 2004, pp. 1227–1237.

[7] G. J. Woeginger, *Exact algorithms for NP-hard problems: A survey*, *Combinatorial Optimization – Eureka, You Shrink*, Springer, LNCS vol. 2570, 2003, pp. 185–207.

Competition structures, hamiltonian digraphs and τ-cycle factors

Hanns-Martin Teichert
(joint work with Martin Sonntag)

All hypergraphs $\mathcal{H} = (V(\mathcal{H}), \mathcal{E}(\mathcal{H}))$, graphs $G = (V(G), E(G))$ and digraphs $D = (V(D), A(D))$ considered here may have isolated vertices but no multiple edges. Loops are allowed only in digraphs.
In 1968 Cohen [2] introduced the *competition graph* $C(D)$ associated with a digraph $D = (V, A)$ representing a food web of an ecosystem. $C(D) = (V, E)$ is the graph with the same vertex set as D (corresponding to the species) and

$$E = \{\{u, w\} \mid u \neq w \ \wedge \ \exists\, v \in V : (u, v) \in A \ \wedge \ (w, v) \in A\},$$

i.e. $\{u, w\} \in E$ if and only if u and w compete for a common prey $v \in V$.
In our paper [8] it is shown that in many cases competition hypergraphs yield a more detailed description of the predation relations among the species in $D = (V, A)$ than competition graphs. If $D = (V, A)$ is a digraph its *competition hypergraph* $C\mathcal{H}(D) = (V, \mathcal{E})$ has the vertex set V and $e \subseteq V$ is an edge of $C\mathcal{H}(D)$ iff $|e| \geq 2$ and there is a vertex $v \in V$, such that $e = \{w \in V \mid (w, v) \in A\}$.
For a graph G, let us call a collection $\{C_1, \ldots, C_p\}$ an *edge clique cover* of G, if each $C_i \subseteq V(G)$ generates a clique in G (not necessarily maximal) or $C_i = \emptyset$, and every edge of G is contained in at least one of these cliques.
If $M = (m_{ij})$ is the adjacency matrix of digraph D, then the competition graph $C(D)$ is the row graph $RG(M)$ (see Greenberg, Lundgren and Maybee [5]). To find a similar characterization for competition hypergraphs, we defined in [8] the *row hypergraph* $R\mathcal{H}(M)$. The vertices of this hypergraph correspond to the rows of M, i.e. to the vertices $v_1, v_2, \ldots, v_n$ of D, and the edges correspond to certain columns; in detail:

$$\mathcal{E}(R\mathcal{H}(M)) = \Big\{\{v_{i_1}, \ldots, v_{i_k}\} | k \geq 2 \wedge \exists j \in \{1, \ldots, n\} : m_{ij} = 1 \Leftrightarrow i \in \{i_1, \ldots, i_k\}\Big\}.$$

Lemma 1 ([8]). . *Let D be a digraph with adjacency matrix M. Then the competition hypergraph $C\mathcal{H}(D)$ is the row hypergraph $R\mathcal{H}(M)$.*

Results for competition graphs of hamiltonian digraphs are given in Fraughnaugh et al. [4].

Theorem 2 ([4]). *A graph G with n vertices is a competition graph of a hamiltonian digraph without loops if and only if G has an edge clique cover $\{C_1, C_2, \ldots, C_n\}$ with a system of distinct representatives $\{v_n, v_1, \ldots, v_{n-1}\}$ such that*

$$(1) \qquad \forall i \in \{1, \ldots, n\} \ : \ v_i \notin C_i.$$

In the same paper [4] it is shown that condition (1) may be omitted in Theorem 2 if D may have loops, Guichard [6] had success in combining both results if G has $n \geq 3$ vertices.

Theorem 3 ([6]). *A graph G with $n \geq 3$ vertices is a competition graph of a hamiltonian digraph without loops if and only if G has an edge clique cover $\{C_1, \ldots, C_n\}$ with a system of distinct representatives.*

Next we provide some results characterizing competition graphs and competition hypergraphs which are important for our further investigations.
A graph G with n vertices is the competition graph of digraph which may have loops if and only if there is an edge clique cover of G containing at most n cliques (cf. Dutton and Brigham [3]). Moreover, if additionaly $G \neq K_2$ is fulfilled, G is even the competition graph of a digraph without loops (cf. Roberts and Steif [7]). Hence the conditions in Theorem 2 and Theorem 3 provide that G is the competition graph of a digraph which may have loops and a digraph without loops, respectively. This is one reason that the additional condition (1) of Theorem 2 may be omitted in Theorem 3.
For hypergraphs the following results are known.

Theorem 4 ([8]). *A hypergraph $\mathcal{H}$ with n vertices is a competition hypergraph of a digraph which may have loops if and only if $|\mathcal{E}(\mathcal{H})| \leq n$.*

Because of the numerous possibilities for edge cardinalities in hypergraphs, the result for digraphs without loops becomes more complicated. For $t \in$ IN we define

$$\mathcal{M}_k = \{M_k \subseteq \{1, \ldots, t\} |\ |M_k| = k\} \quad \text{for} \quad k = 1, \ldots, t.$$

Theorem 5 ([8]). *Let $\mathcal{H}$ be a hypergraph with n vertices and $\mathcal{E}(\mathcal{H}) = \{e_1, \ldots, e_t\}$. Then $\mathcal{H}$ is a competition hypergraph of a digraph without loops if and only if*

$$\forall k \in \{1, \ldots, t\}\ \forall M_k \in \mathcal{M}_k : |\bigcap_{j \in M_k} e_j| \leq n - k.$$

There are two interesting points of view for the investigations of competition hypergraphs of hamiltonian digraphs:

a) The $t \leq n$ edges of the competition hypergraph $C\mathcal{H}(D)$ correspond to certain cliques of a suitable edge clique cover $\{C_1, \ldots, C_n\}$ of the competition graph $C(D)$. In Theorem 2 and 3 there are conditions for all these n cliques $C_1, \ldots, C_n$. In case of hypergraphs it would be desirable to formulate conditions only for the $t \leq n$ edges, and this will be possible.

b) Considering Theorems 2 and 3 the question arises whether in case of loopless digraphs a condition corresponding to (1) is needed or not?

Our results will show that, unfortunately, the answer to the question b) is yes. However, if we do not postulate such a condition we can prove a weaker result; this motivates the following definition. According to Bang-Jensen and Gutin [1] a system $\{\vec{c}_1, \ldots, \vec{c}_\tau\}$ of oriented cycles in a digraph D is called a *τ-cycle factor* if every vertex of D is contained in exactly one cycle $\vec{c}_j \in \{\vec{c}_1, \ldots, \vec{c}_\tau\}$. Clearly, for $\tau = 1$ the digraph D is hamiltonian.

The following result characterizes competition hypergraphs of digraphs D having a τ-cycle factor. Note that sometimes $\tau \geq 2$ is unavoidable.

Theorem 6 ([9])**.** *Let $\mathcal{H}$ be a hypergraph with n vertices and $\mathcal{E}(\mathcal{H}) = \{e_1, \ldots, e_t\}$. Then the following conditions are equivalent.*

(i) $\mathcal{H}$ is the competition hypergraph of a loopless digraph D having a τ-cycle factor.

(ii) $\forall\ k \in \{1, \ldots, t\}\ \forall M_k \in \mathcal{M}_k : \big| \bigcup_{j \in M_k} e_j \big| \ \geq k \ \wedge \ \big| \bigcap_{j \in M_k} e_j \big| \ \leq n - k.$

(iii) $\mathcal{H}$ is the competition hypergraph of a loopless digraph and $\mathcal{E}(\mathcal{H})$ has a system of distinct representatives.

For the proof of Theorem 6 we use Lemma 1, Theorem 5, Halls Theorem and the fact that any permutation of rows or columns in a matrix M does not change the row hypergraph $R\mathcal{H}(M)$ (up to isomorphism).

Finally we characterize the competition hypergraphs of hamiltonian digraphs without loops; for this purpose the condition (2) below, which corresponds to (1) in Theorem 2, is necessary.

Theorem 7 ([9])**.** *A hypergraph $\mathcal{H}$ with $V(\mathcal{H}) = \{v_1, \ldots, v_n\}$ and $\mathcal{E}(\mathcal{H}) = \{e_1, \ldots, e_t\}$ is the competition hypergraph of a hamiltonian digraph without loops if and only if there is a subset $\{j_1, \ldots, j_t\} \subseteq \{1, \ldots, n\}$ of pairwise distinct indices such that $\{v_{j_1-1}, \ldots, v_{j_t-1}\}$ (indices taken mod n) is a system of distinct representatives of $\{e_1, \ldots, e_t\}$ and furthermore*

$$\forall i \in \{1, \ldots, t\} : v_{j_i} \notin e_i. \tag{2}$$

As an immediate consequence we obtain a characterization for the case of hamiltonian digraphs with loops allowed.

Corollary 8 ([9])**.** *A hypergraph $\mathcal{H}$ is the competition hypergraph of a hamiltonian digraph (which may have loops) if and only if $\mathcal{E}(\mathcal{H})$ has a system of distinct representatives.*

References

[1] J. Bang-Jensen and G. Gutin, *Digraphs: theory, algorithms and applications*, Springer, London, 2001.

[2] J.E. Cohen, *Interval graphs and food webs: a finding and a problem*, Rand Corporation Document 17696-PR, Santa Monica, CA, 1968.

[3] R.D. Dutton and R.C. Brigham, *A characterization of competition graphs*, Discr. Appl. Math. **6** (1983), 315–317.

[4] K.F. Fraughnaugh, J.R. Lundgren, S.K. Merz, J.S. Maybee and N.J. Pullman, *Competition graphs of strongly connected and hamiltonian digraphs*, SIAM J. Discr. Math. **8** (1995), 179–185.

[5] H.J. Greenberg, J.R. Lundgren and J.S. Maybee, *Inverting graphs of rectangular matrices*, Discr. Appl. Math. **8** (1984), 255–265.

[6] D.R. Guichard, *Competition graphs of hamiltonian digraphs*, SIAM J. Discr. Math. **11** (1998), 128–134.
[7] F.S. Roberts and J.E. Steif, *A characterization of competition graphs of arbitrary digraphs*, Discr. Appl. Math. **6** (1983), 323–326.
[8] M. Sonntag, H.-M. Teichert, *Competition hypergraphs*, Discr. Appl. Math. **143** (2004), 324–329.
[9] M. Sonntag, H.-M. Teichert, *Competition hypergraphs of digraphs with certain properties*, (2006), to appear.

Page migration in dynamic networks

Friedhelm Meyer auf der Heide
(joint work with Marcin Bienkowski)

Page migration is a basic, simple model for data management in networks: A page of size D is given that has to be stored in exactly one node of a given network. The task is to serve a sequence of requests to data items from the page where each request is described by the issuing processor. The cost of serving the request is the shortest path length between the issuing processor and the page. The page migration algorithm has to decide, after each served request, if the page is migrated, and, in case of "yes", where it is migrated to. The cost for migration is D times the migration distance. As the shortest-path distances in a network form a metric, we will from now on take the more general view and assume nodes and distances to be defined by some metric space.
In the online setting, each decision - migrate or not - has to be done without knowledge about future requests. The quality of such an online algorithm is measured by its competitive ratio, i.e., the factor by which the online algorithm is more expensive than an optimal offline algorithm that knows the whole input sequence in advance. Page migration is a very well studied online problem, constant competitive ratio can be achieved, see [1, 2, 3, 4, 14]. Further, more general data management problems (bounded memory, multiple copies of data items,...) were investigated, see [5, 12, 13].

The dynamic page migration (DPM) problem.
In this talk, we extend page migration to dynamic networks, where a further input stream dictates network changes. We assume a "speed limit" on the nodes, i.e., each node can move only within a cycle of diameter one in each step. Like in the static case, the offline dynamic page migration (DPM) problem can be solved by an easy dynamic programming approach in polynomial time.
Since the input sequence consists of two streams, one describing the request patterns and one reflecting the changes in network topology, it is reasonable to assume that they are created by two separate adversarial entities, the request adversary and the network adversary. This separation yields different scenarios depending on ways in which these adversaries may act and interact.

The adversarial (cooperative) scenario.
The most straightforward model, which also creates the most difficult task to solve, arises when both adversaries may cooperate to create the combined input sequence. For this scenario, in [8] we construct a deterministic strategy which is $O(\min\{n \cdot \sqrt{D}, D\})$-competitive. This algorithm is up to a constant factor optimal, due to the matching lower bound for adaptive-online adversaries, given in [10]. Further, we show how to randomize this strategy to get a competitive ratio of $O(\sqrt{D} \cdot \log n, D\})$ against an oblivious (may not see outcomes of coin flips) adversary. This result is up to a $O(\sqrt{\log n})$ factor optimal as shown in [8].
As shown above, the competitive ratios of the best possible algorithms for the DPM problem are large, even against the weakest, the oblivious adversaries. It can be inferred that the poor performance of algorithms for this scenario is caused by the fact that the network and request adversaries might combine their efforts in order to destroy our algorithm. However, it is not an easy task to model such a non-cooperative behavior. Therefore, it was proposed in [6, 9, 10] that the DPM problem could be analyzed in another extreme case, where one of the adversaries is replaced by a stochastic process. This leads to the following two scenarios.

The Brownian motion scenario.
In this scenario the mobile nodes perform a random walk on a bounded area of diameter B, and the request adversary dictates which nodes issue requests during runtime. However, the adversary is “oblivious”, i.e. it has to create the whole request sequence $(\sigma_t)_t$ in advance, without knowledge of the actual configuration sequence $(c_t)_t$ induced by a random walk. The definition of competitiveness has to be adapted appropriately to reflect the fact that the input sequence is created both by an adversary and a stochastic process. A deterministic algorithm ALG is R-competitive with probability p, if there exists a constant A, s.t., for all request sequences $(\sigma_t)_t$, it holds that

$$(1) \qquad \Pr_{(c_t)_t}\Big[C_ALG((c_t)_t,(\sigma_t)_t) \le R \cdot C_OPT((c_t)_t,(\sigma_t)_t) + A\Big] \ge p \ ,$$

where the probability is taken over all possible configuration sequences generated by the random movement.
The main result of [9], based on the preliminary result of [10],is an algorithm MAJ, which is $O(\min\{\sqrt[4]{D}, n\} \cdot polylog(B, D, n))$-competitive. This result holds for 1-dimensional areas if $B \le \tilde{O}(\sqrt{D})$, or for any constant-dimensional areas if $B \ge \tilde{O}(\sqrt{D})$. The ratio is achieved w.h.p., i.e. , the probability p occurring in (1) can be amplified to $1 - D^{-\alpha}$ by setting $A = \alpha \cdot A_0$ for a fixed constant A_0.

The stochastic requests scenario.
In this scenario. it is assumed that requests appear with some given frequencies, i.e. , in step t, σ_t is a node chosen randomly according to a fixed probability distribution π. Analogously, a deterministic algorithm ALG is R-competitive with probability p, if there exists a constant A, s.t. for all possible network topology changes (configuration sequences) $(c_t)_t$ and all possible probability distributions

π holds

$$\mathrm{Pr}_{(\sigma_t)_t}\left[C_ALG((c_t)_t,(\sigma_t)_t) \leq R \cdot C_OPT((c_t)_t,(\sigma_t)_t) + A\right] \geq p \ , \tag{2}$$

where the probability is taken over all possible request sequences $(\sigma_t)_t$ generated according to π.
The Move-To-First-Request algorithm presented in [6] achieves $O(1)$-competitive ratio, w.h.p. In this context, high probability means that one can achieve probability $1 - D^{-\alpha}$, if the input sequence is sufficiently long. Moreover, the algorithm can be slightly modified to handle also the following cost function

$$c_t(v_a, v_b) = (d_t(v_a, v_b))^\beta + 1 \ , \tag{3}$$

for any constant β, still remaining $O(1)$-competitive. For the case of wireless radio networks, one can choose the parameter β to respect a *propagation exponent* of the medium. For example by setting $\beta = 2$, the cost definition reflects the energy consumption used the send the message in the ideally free space along a given distance. Thus, this result minimizes, up to a constant factor, the total energy used in the system.

References

[1] D. Achlioptas, M. Chrobak, and J. Noga. Competitive analysis of randomized paging algorithms. *Theoretical Computer Science*, 234(1–2):203–218, 2000.

[2] B. Awerbuch, Y. Bartal, and A. Fiat. Distributed paging for general networks. *Journal of Algorithms*, 28(1):67–104, 1998. Also appeared in *Proc. of the 7th SODA*, pages 574–583, 1996.

[3] Y. Bartal. Distributed paging. In *Dagstuhl Workshop on On-line Algorithms*, pages 97–117, 1996.

[4] Y. Bartal, M. Charikar, and P. Indyk. On page migration and other relaxed task systems. *Theoretical Computer Science*, 268(1):43–66, 2001. Also appeared in *Proc. of the 8th SODA*, pages 43–52, 1997.

[5] Y. Bartal, A. Fiat, and Y. Rabani. Competitive algorithms for distributed data management. *Journal of Computer and System Sciences*, 51(3):341–358, 1995. Also appeared in *Proc. of the 24nd STOC*, pages 39–50, 1992.

[6] M. Bienkowski. Dynamic page migration with stochastic requests. In *Proc. of the 17th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, 2005.

[7] M. Bienkowski and F. Meyer auf der Heide. Page migration in dynamic networks. In *Proc. of the 30th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 1–14, 2005.

[8] M. Bienkowski, M. Dynia, and M. Korzeniowski. Improved algorithms for dynamic page migration. In *Proc. of the 22nd Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 365–376, 2005.

[9] M. Bienkowski and M. Korzeniowski. Dynamic page migration under brownian motion. In *Proc. of the European Conf. in Parallel Processing (Euro-Par)*, 2005. To appear.

[10] M. Bienkowski, M. Korzeniowski, and F. Meyer auf der Heide. Fighting against two adversaries: Page migration in dynamic networks. In *Proc. of the 16th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 64–73, 2004.

[11] C. Lund, N. Reingold, J. Westbrook, and D. C. K. Yan. Competitive on-line algorithms for distributed data management. *SIAM Journal on Computing*, 28(3):1086–1111, 1999. Also appeared as On-Line Distributed Data Management in *Proc. of the 2nd ESA*, pages 202–214, 1994.

[12] B. M. Maggs, F. Meyer auf der Heide, B. Vöcking, and M. Westermann. Exploiting locality for data management in systems of limited bandwidth. In *Proc. of the 38th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 284–293, 1997.

[13] F. Meyer auf der Heide, B. Vöcking, and M. Westermann. Caching in networks. In *Proc. of the 11th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 430–439, 2000.

[14] J. Westbrook. Randomized algorithms for multiprocessor page migration. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 7:135–150, 1992.

Approximation algorithms for scheduling malleable tasks under general precedence constraints

Klaus Jansen

(joint work with Hu Zhang)

In this talk we study the problem of scheduling malleable tasks with precedence constraints. We are given m identical processors and n tasks. For each task the processing time is a function of the number of processors alloted to it. In addition, the tasks must be processed according to the precedence constraints. The goal is to minimize the makespan (maximum completion time) of the resulting schedule. The best previous approximation algorithm by Lepere, Trystram, and Woeginger [1] has a ratio $3 + \sqrt{5} \approx 5.236$. We develop an improved approximation algorithm with a ratio at most 4.7306 [2]. In addition we study the important case where the speedup function is concave in the number of processors. In this case the work function of any malleable task is convex in the processing time. We propose for this case an approximation algorithm with improved ratio 3.2919 [3].

References

[1] R. Lepere, D. Trystram and G. Woeginger, *Approximation algorithms for scheduling malleable tasks under precedence constraints*, International Journal of Foundations of Computer Science 13, 2002, 613-627.

[2] K. Jansen, H. Zhang, *An approximation algorithm for scheduling malleable tasks under general precedence constraints*, Proceedings of the 16th International Symposium on Algorithms and Computation, ISAAC 2005, 236-245.

[3] K. Jansen, H. Zhang, *Scheduling malleable tasks with precedence constraints*, Proceedings of the 17th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2005, 86-95.

Two remarks on coloring graphs without long induced paths

Van Bang Le

(joint work with Bert Randerath, Ingo Schiermeyer)

Given an integer $k \geq 1$, a k-coloring of a graph is an assigment of k colors $1, \ldots, k$, to the vertices of the graph such that adjacent vertices receive different colors. Given a graph G, the k-Coloring problem asks whether G admits a k-coloring. A path on t vertices is denoted by P_t. Graphs without induced P_t are called P_t-free.

For $t \leq 3$, k-Coloring on P_t-free graphs is trivial. On P_4-free graphs (also called *cographs*), it is well-known that k-Coloring is solvable in linear time (see, e.g., [1].) Recently, the following results have been obtained:

Theorem 1 ([5, 2])**.** 3-Coloring *is solvable in polynomial time on P_5-free graphs.*

Indeed, 3-Coloring can be solved on P_5-free graphs in time $O(n^\alpha)$; see the recent survey paper [3] for more information (n and m are the vertex, respectively, edge number of the input graphs, $O(n^\alpha)$ is the time needed to perform an $n \times n$ matrix multiplication; currently, $\alpha \approx 2.376$.)

Theorem 2 ([4])**.** 3-Coloring *is solvable in time $O(mn^\alpha)$ on P_6-free graphs.*

Theorem 3 ([5])**.** 4-Coloring *is NP-complete on P_{12}-free graphs, and* 5-Coloring *is NP-complete on P_8-free graphs.*

My first remark is, by modifying a reduction given in [5], the first part in Theorem 3 can be improved to:

Theorem 4. 4-Coloring *is NP-complete on P_9-free graphs.*

Hence, for all $k \geq 4$ and all $t \geq 9$, k-Coloring is NP-complete on P_t-free graphs. The computational complexity of 4-Coloring, however, is still unknown on P_5-free graphs.
My second remark is that a first nontrivial result in this direction can be obtained as follows:

Theorem 5. 4-Coloring *is solvable in polynomial time on (P_5, C_5)-free graphs.*

Where C_ℓ denotes the cycle on ℓ vertices. The main part of the proof of Theorem 5 is when the graph G in question contains an induced $\overline{C_7}$. In this case, deciding if G is 4-colorable can be polynomially reduced to deciding if a P_5-free graph is 3-colorable, and if a certain 2Sat instance can be satisfied.
Below is the current status of the computational complexity of k-Coloring on P_t-free graphs.

$k \setminus t$	4	5	6	7	8	9	10	11	...
3	LIN	$O(n^\alpha)$	$O(mn^\alpha)$	?	?	?	?	?	...
4	LIN	?	?	?	?	NP-c	NP-c	NP-c	...
5	LIN	?	?	?	NP-c	NP-c	NP-c	NP-c	...
6	LIN	?	?	?	NP-c	NP-c	NP-c	NP-c	...
7	LIN	?	?	?	NP-c	NP-c	NP-c	NP-c	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 1. LIN, NP-c, '?' means that the complexity status of the corresponding k-Coloring problem on P_t-free graphs is linear, NP-complete, unknown, respectively.

References

[1] D.G. Corneil, Y. Perl, and L.K. Stewart, *Cographs: recognition, applications, and algorithms*, Congressus Numer. **43** (1984), 249–258.

[2] B. Randerath, I. Schiermeyer, M. Tewes, *Three-colourability and forbidden subgraphs.II: polynomial algorithms*, Discrete Math. **251** (2002), 137–153.

[3] B. Randerath, I. Schiermeyer, 3-*Colorability in P for P_6-free graphs*, Discrete Appl. Math. **136** (2004), 299–313.

[4] B. Randerath, I. Schiermeyer, *Vertex coloring and forbidden subgraphs – A survey*, Graphs Combinatorics **20** (2004), 1–40.

[5] G.J. Woeginger, J. Sgall, *The complexity of coloring graphs without long induced paths*, Acta Cybern. **15** (2001), 107–117.

Precoloring extensions distance constraints for list colorings

Margit Voigt

1. Problem

Let G be a simple graph and $L(v)$ a set of available colors for every $v \in V$. The set $L(v)$ is also called a *list* of v and the collection of all lists is called a list assignment L of G. The graph G is *L-list colorable* if for a given L a proper coloring of the vertices exists where every vertex gets a color from its list.
Now we assume additionally that there is a subset $W \subseteq V$ of the vertex set which is already precolored. Denote by $d(W)$ the minimum distance between two components of W in G. We would like to extend the precoloring of W to a proper coloring of the whole vertex set. Clearly the existence of such an extension depends on $d(W)$ and the number of available colors for an ordinary coloring or the length of the lists of the list assignment, respectively. First results in this direction where given by Albertson 1998 [1] answering a question of Thomassen from 1997. He stated that if G is k-colorable and W is independent with $d(W) \geq 4$ then every (k+1)-coloring of W can be extended to a proper $(k+1)$-coloring of V. In the last years there are several papers, [2] - [8] and [11], dealing with this topic from different points of view.

2. Extension of Brooks' Theorem

Let us consider simple graphs $G = (V, E)$ with maximum degree $k = \Delta(G) \geq 3$. The well-known theorem of Brooks [9] states that such a graph is k-colorable if it does not contain K_{k+1} as a component. This theorem can be generalized in several directions. Among others in [10, 12] a Brooks-type theorem is proved saying that a graph G with maximum degree $k = \Delta(G) \geq 3$ is L-list colorable for every k-assignment L if G does not contain K_{k+1} as a component.
Here we ask for the extension of a precoloring of W to a proper list coloring if every vertex has a list of $k = \Delta(G)$ colors. Axenovich [8] and Albertson, Kostochka and West [5] proved that for independent W, $k = \Delta(G) \geq 3$ and $d(W) \geq 8$

such an extension is always possible if G does not contain a K_{k+1} as subgraph. Furthermore they give an example showing that the bound 8 is sharp. Remarkably, the mentioned example is a 1-connected graph. For 2-connected graphs the bound $d(W)$ is improved in [15] and [16].

Theorem 1. *If $G = (V, E)$ is 2-connected, $|L(v)| = \Delta \geq 4$ $\forall v \in V$ and $d(W) \geq 4$ then every L-list coloring of W can be extended to an L-list coloring of V.*

Theorem 2. *If $G = (V, E)$ is 2-connected, $|L(v)| = \Delta = 3$ $\forall v \in V$ and $d(W) \geq 6$ then every L-list coloring of W can be extended to an L-list coloring of V.*

Moreover there are given examples showing that the above bounds are sharp. The main tool for proving the theorems is the investigation of an L-list assignment with $|L(v)| \geq d(v)$ where $d(v)$ is the degree of v in G. It is pointed out that there is a polynomial algorithm which decides whether an arbitrary graph G is L-list colorable for such an assignment. If there is an L-list coloring then the algorithm finds such a coloring. Obviously we can also apply this algorithm to find list coloring extensions mentioned in the theorems.

3. Forbidden minors

Let G be a graph not containing K_{k+1} as minor with $\chi(G) = k$ and assume that $G[W]$ is s-colorable.
Hutchinson and Moore [11] gave the following bounds for d, where d is the minimum $d(W)$ such that every $(k + s - 1)$- coloring of W (each component of $G[W]$ is s-colored) extends to a $(k + s - 1)$- (ordinary) coloring of all of G.

$k \backslash s$	2	3	4	5
2	5	–	–	–
3	7, 8	5	–	–
4	7, 8	7	6	–
5	7, 8	7, 8	7	6

Dealing with the problem $k = 3, s = 2$ we investigated the subclass of outerplanar graphs and obtained in [14] the following result for list colorings.

Theorem 3. *Let G be an outerplanar graph, $G[W]$ is bipartite, $d(W) \geq 7$ and $|L(v)| = 4$ for all $v \in V \setminus W$. Then every coloring of W, where each component of $G[W]$ is two-colored, extends to a proper list coloring of V.*

Moreover there is a polynomial algorithm to find such an extension.

4. Planar bipartite graphs

Assume that G is a planar bipartite graph and $|L(v)| \geq 3$ for all $v \in V$. The problem is to decide whether a precoloring of W extends to a proper list coloring of V. If $|W|$ is at most 2 then a precoloring is always extendable and the list coloring can be found in polynomial time. The problem is open for W with at least 3 vertices. However there are some related results. From a result of Kratochvíl

and Tuza [13] follows that the mentioned extension problem is NP-complete if W is independent and $d(W) \geq 4$.
Question: *What happens if* W *is independent and* $d(W) \geq 5$*?*
A related result dealing with ordinary colorings without requirement of planarity is proved in [11] (table given above, $k = s = 2$).

References

[1] M.O.Albertson, You can't paint yourself into a corner, J. Combinatorial Theory Ser. B, 73 (1998), 189-194
[2] M.O.Albertson, J.P. Hutchinson, Extending colorings of locally planar graphs, J. Graph Theory 36 (2001), 105-116
[3] M.O.Albertson, J.P. Hutchinson, Graph color extensions: when Hadwiger's conjecture and embeddings help, Elect. J. Comb. 9 (2002), # R18
[4] M.O.Albertson, J.P. Hutchinson, Extending precolorings of subgraphs of locally planar graphs, European J. Comb., 25 (2004), 863-871
[5] M.O. Albertson, A.V.Kostochka, D.B.West, Precoloring extension of Brooks' Theorem, SIAM J. Discr. Math. 18 (2004), 542-553
[6] M.O.Albertson, E.H.Moore, Extending graph colorings, J. Combinatorial Theory Ser. B, 77(1999), 83-95
[7] M.O.Albertson, E.H.Moore, Extending graph colorings using no extra colors, Discrete Math. 234 (2001), 125-132
[8] M.Axenovich, A note on graph coloring extensions and list colorings, Electronic J. Comb. 10 (2003) #N1
[9] R.L.Brooks, On colouring the nodes of a network, Proc. Cambridge Philos. Soc. 37 (1941), 194-197
[10] P.Erdős, A.L.Rubin, H.Taylor, Coosability in graphs, Proc. West Coast Conference on Combinatorics, Graph Theory and Computing (Humboldt State Univ., Arcata, Calif. 1979), Comgress. Numer. XXVI (Utilitas Math., Winnipeg, 1980), 125-157
[11] J.P. Hutchinson, E.H. Moore, Distance Constraints in Graph Color Extensions, manuscript, 2005
[12] A.V.Kostochka, M.Stiebitz, B.Wirth, The colour theorems of Brook's and Gallai extended, Discrete Math. 162 (1996), 299-303
[13] J. Kratochvíl, Zs. Tuza, Algorithmic complexity of list colorings, Discrete Applied Math. 50 (1994), 297-302
[14] A. Pruchnewski, M. Voigt, Precoloring extension for outerplanar graphs, manuscript 2005
[15] M. Voigt, Precoloring extension for 2-connected graphs, submitted, SIAM J. Discr. Math.
[16] M. Voigt, Precoloring extension for 2-connected graphs with $\Delta(G) = 3$, manuscript 2005

On Exponential Time Algorithms for Treewidth

Hans L. Bodlaender
(joint work with Fedor V. Fomin, Arie Koster, Dieter Kratsch, Dimitrios M. Thilikos)

In this talk, algorithms that compute the treewidth of graphs are discussed. The main results are twofold. First, we give a dynamic programming algorithm that computes the treewidth of a graph with n vertices in $O^*(2^n)$ time. (The O^*-notation suppresses polynomial factors.) The dynamic programming algorithm is based upon a characterization of treewidth in terms of a cost measure associated to linear orderings (inspired by the fact that treewidth can be written as the minimum over all chordal supergraphs of the maximum clique size, and that chordal graphs have a perfect elimination scheme.) The resulting algorithm has a structure that is similar to the famous Held-Karp dynamic programming algorithm for treewidth [2].

The dynamic programming algorithm is *not* a theoretical improvement on existing algorithms: the algorithm of Fomin et al [1], and the recent improvement on it by Villanger [3] have a smaller constant at the base of the exponent. However, the dynamic programming algorithm is easy to implement, and experiments show that it works well for graphs with sizes between 30 and 80 vertices.

The experiments also show that the use of memory is often a bottleneck: the algorithm also uses worst case $\Theta(2^n)$ time; when the data for the program do no longer fit into main memory, the program becomes very slow, spending most of its time on obtaining data from and writing data to secondary memory. Thus, our second result deals with exponential time algorithms for treewidth with polynomially bounded memory. We give a recursive algorithm that computes the treewidth of a graph exactly, and that uses $O^*(2.9978^n)$ time and polynomial memory. Our algorithm is based upon a divide and conquer approach, and uses results from the theory of potential maximal cliques, and the existence of balanced separators of the size the treewidth of the graph.

Similar algorithms exist also for related problems, e.g., the same time bounds can be obtained for the minimum fill-in problem.

References

[1] F. V. Fomin, D. Kratsch, and I. Todinca. Exact (exponential) algorithms for treewidth and minimum fill-in. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming*, pages 568–580, 2004.

[2] M. Held and R. Karp. A dynamic programming approach to sequencing problems. *J. SIAM*, 10:196–210, 1962.

[3] Y. Villanger. Counting and listing all potential maximal cliques of a graph. Reports in Informatics 302, Department of Informatics, University of Bergen, Bergen, Norway, 2005.

Parallel Matching Algorithms

Stefan Hougardy
(joint work with D. Vinkemeier)

1. Introduction

The class NC is the class of all problems that are computable in polylogarithmic time with polynomially many processors. Uehara and Chen [8] have shown that there exists an NC approximation algorithm for the weighted matching problem that achieves an approximation ratio of $\frac{1}{2}-\epsilon$. We improve this result by presenting a $1-\epsilon$ NC approximation algorithm for the weighted matching problem.

2. Preliminaries

A *matching* M in a graph $G = (V, E)$ is a subset of the edges E of G such that no two edges in M have a vertex in common. Let $G = (V, E)$ be a graph and $w : E \to \mathbb{R}_+$ be a function which assigns a positive weight to each of the edges of G. Then the weight $w(F)$ of a subset $F \subseteq E$ of the edges of G is defined as $w(F) := \sum_{e \in F} w(e)$. The weighted matching problem is to find a matching M in G that has maximum weight.

The model of computation we use is the CREW PRAM (concurrent reads exclusive writes parallel random access machine). In this model there exists a sequence of indexed random access machines, each of which knows its own index. The processors synchronously execute the same central program, communicating with one another through a shared random access memory. CREW means that several processors can concurrently read a particular memory address but at most one processor can write to a single memory address in each step. See [6, 7] for more background on parallel algorithms.

The quality of an approximation algorithm for the weighted matching problem is measured by its so-called *approximation ratio*. An approximation algorithm has an approximation ratio of c, if for all graphs it finds a matching with a weight of at least c times the weight of an optimal solution.

3. The Ranked Augmentation Graph

The main idea of our algorithm is to start with some (possibly empty) matching and to allow each processor to make some local changes of this matching to improve its weight. The local changes made by different processors need to be independent of each other. To achieve this we construct from the given graph a new graph which we call the augmentation graph. In this augmentation graph an independent set of vertices corresponds to a set of pairwise independent local changes of the matching in the given graph. As we aim to increase the weight of a given matching by a set of local changes by as much as possible, we will *rank* all possible local changes for a given matching. This means that we partition the set of all possible local changes of a matching into classes such that the local changes within each class achieve a similar increase in weight. Instead of computing an independent set in the augmentation graph the idea is to compute an independent set in each of the classes of the partition. The augmentation graph together with the ranking of its vertices will be called the ranked augmentation graph. We are going to describe the construction of this graph in more detail in the following.

Given a graph $G = (V, E)$ and a matching $M \subseteq E$ let an *augmentation with respect to the matching* M be any matching $S \subset E \setminus M$. If the matching M is known from the context we will simply say that S is an augmentation. Let $M(S)$ denote all edges in M that have a vertex in common with an edge in S. Then $(M \setminus M(S)) \cup S$ is again a matching. We say that $(M \setminus M(S)) \cup S$ is the matching that is obtained by augmenting M by S. If S is an augmentation with respect to M then the *gain* of augmenting M by S is defined as $gain_M(S) = w(S) - w(M(S))$. Thus the gain is the difference of weight between the matching M and the matching

$(M \setminus M(S)) \cup S$. Our definition allows augmentations that have negative gain, but our algorithm will only consider augmentations that have positive gain, i.e., that increase the weight of the matching M.
The *size* of an augmentation S is simply the number of edges contained in S. Let $G = (V, E)$ be a graph, $M \subseteq E$ be a matching and $l > 0$ be an integer. Then the *augmentation graph* $G' = G'(G, M, l)$ is defined as follows. The vertices of G' are all augmentations with respect to M of size at most l. Two such vertices are connected by an edge if the corresponding augmentations have at least one vertex of G in common.
All augmentations which are vertices of G' will be ranked according to their gains as follows (this is very similar to the ranking of the edges that Uehara and Chen [8] used in their algorithm). First find the augmentation of $V(G')$ with the largest gain denoted by $gain_{max}$. For each vertex S in $V(G')$ we define its *rank* $r(S)$ as follows (n denotes the number of vertices in G)

- If $gain(S) \le \frac{gain_{max}}{l \cdot n}$ then $r(S) = 0$
- Otherwise $r(s) = i > 0$ where i is the smallest integer for which it is true that $gain(S) \le 2^i \cdot \frac{gain_{max}}{l \cdot n}$.

This definition implies that for constant l the rank of an augmentation is an integer of size $O(\log n)$.
We call the augmentation graph G' together with the ranking of its vertices the *ranked augmentation graph* of G. This graph can be computed using $O(n^{4l})$ processors and $O(\log n)$ time.

4. The Algorithm

Our algorithm for computing a $1-\epsilon$ approximation of a maximum weight matching starts with the empty matching and makes c calls to the algorithm `ImproveMatching` for some constant c which depends only on ϵ. The algorithm `ImproveMatching` is shown in Figure 1. This algorithm takes as input a weighted graph G and a matching M and returns a new matching M'. It starts by calculating out of G and M the ranked augmentation graph G' as described in Section 3. Within the graph G' a maximal independent set is calculated in the following way. Let V_i be the vertices of G' that have rank i. Then starting from the highest rank a maximal independent set ALG_i is calculated in the subgraph of G' that is induced by the vertices of V_i which have not yet been removed from G' (using for example the algorithm from [5]). All neighbors of vertices of ALG_i are removed to ensure that the union of all sets ALG_i is an independent set of G'. The process considers all vertices from the highest rank down to those of rank 1. Vertices of rank 0 are thrown away. The set ALG is the union of all the sets ALG_i and is by construction a maximal independent set in $G' \setminus V_0$. ALG is used as an augmenting set for M to obtain the new matching M' which is returned by the algorithm `ImproveMatching`.
The main statement about the weight of the matching M' that is returned by the algorithm `ImproveMatching` is as follows:

```
ImproveMatching: $G = (V, E), w : E \to R_+$, matching $M$
Output: matching $M'$

1   ALG = $\emptyset$
2   calculate the ranked augmentation graph $G'$
3   for $i = rank_{max}$ downto 1 do
4       calculate a maximal independent set $ALG_i$ in the graph $G'_i := (V'_i, E'_i)$
        that is induced by all vertices still in $G'$ that have rank $i$
5       remove all vertices from $G'$ that have neighbors in $ALG_i$
6       $ALG = ALG \cup ALG_i$
7   $M' = M$ augmented by all augmentations in $ALG$
```

FIGURE 1. An NC algorithm for improving the weight of a matching M.

Theorem 1. *If the algorithm* `ImproveMatching` *gets a matching M as input then it returns a matching M' such that*

$$w(M') \geq w(M) + \frac{1}{4l} \cdot \left(\frac{l-1}{l} \cdot w(M^*) - w(M) \right) .$$

Using this result we can show that we get an NC algorithm that finds a matching of weight at least $(1 - \epsilon) \cdot w(M^*)$ by making a constant number of calls to the algorithm `ImproveMatching`.

Theorem 2. *For every $\epsilon > 0$ there exists an NC algorithm that finds in a weighted graph a matching of weight at least $(1 - \epsilon) \cdot w(M^*)$.*

Our algorithm needs $n^{O(\frac{1}{\epsilon})}$ processors. This is the same amount of processors that is needed in the $1 - \epsilon$ NC-approximation algorithm for the unweighted case [4].

REFERENCES

[1] D.E. Drake, S. Hougardy, Improved linear time approximation algorithms for weighted matchings, In: Approximation, Randomization, and Combinatorial Optimization, (Approx/Random) 2003, S.Arora, K.Jansen, J.D.P.Rolim, A.Sahai (Eds.), LNCS 2764, Springer 2003, 14–23.

[2] D.E. Drake, S. Hougardy, A linear time approximation algorithm for weighted matchings in graphs, ACM Transactions on Algorithms 1 (2005), 107–122.

[3] S. Hougardy, D.E. Vinkemeier, Approximating weighted matchings in parallel, to appear in Information Processing Letters.

[4] T. Fischer, A.V. Goldberg, D.J. Haglin, S. Plotkin, Approximating matchings in parallel, Information Processing Letters 46 (1993), 115–118.

[5] M. Goldberg, T. Spencer, A new parallel algorithm for the maximal independent set problem, SIAM Journal on Computing 18:2 (1989), 419–427.

[6] J. Jájá, An Introduction to Parallel Algorithms, Addison-Wesley, Reading, Massachusetts 1992.

[7] M. Karpinski, W. Rytter, Fast Parallel Algorithms for Graph Matching Problems, Clarendon Press, Oxford 1998.

[8] R. Uehara, Z.-Z. Chen, Parallel approximation algorithms for maximum weighted matching in general graphs, Information Processing Letters 76:1-2 (2000), 13–17.

Learning Wadrop Equilibria

Berthold Vöcking
(joint work with Simon Fischer, Harald Räcke)

Recent contributions in the field of algorithmic game theory have provided much insight into the structure of Nash equilibria for routing in networks that lack central coordination. Prominent results include bounds on the *price of anarchy* measuring the performance loss due to selfishness in relation to the centrally optimized solution [1, 4, 6, 13, 17, 18], and questions regarding how to design networks such that equilibria induced by selfish agents coincide with the globally optimal solution, e. g. by imposing taxes [5, 10] or by introducing a global instance that controls a small fraction of the traffic [11, 16]. These static analyses of Nash equilibria disregard the question of how an equilibrium is actually reached. Classical game theory does also not give an answer to this question. It justifies Nash equilibria with idealistic assumptions like unbounded rationality and global knowledge that, however, are rarely fulfilled in a real-world networks like the Internet.
We study the question of how a large population of agents can *compute* or *learn* an equilibrium efficiently based on simple sampling and adaption policies. Our motivation is twofold. On the one hand, we want to support the previous analyses of Nash equilibria by showing that a population of agents following simple, myopic, and reasonable rules quickly converges to a Nash equilibrium. On the other hand, we think that our analysis may contribute to the design of distributed adaptive rerouting protocols that quickly converge to stable routing allocations. Our study is based on the well known traffic model of Wardrop [19] (see also [18]) in which the traffic is modelled in form of an infinite number of agents each of which responsible for an infinitesimal amount of traffic. We imagine that the agents play a repeated game in rounds. In each round, each agent may compare the latency of his current route with the latency of another route and switch to the other route if it promises a better latency. The problem with this natural approach is that other agents might switch simultaneously to the same route so that the latency of an agent may not improve or even get worse instead of better. This way, the game may get stuck in oscillations. This phenomenon is also well known in the *networks* community and the instabilities due to oscillations observed within the ARPANET project are one of the major reasons why the Internet does not support adaptive routing, see e.g. [12, 14, 15].
In [8], it was shown that such oscillation effects can be avoided by letting the agents sample alternative routes at random and migrate with a probability depending on the observed latency difference. The weakness of the routing protocols presented in [8] is that the migration policy depends heavily on the first derivatives of the latency functions: In order to avoid oscillations the probability to switch to another path is scaled down by a factor that is linear in the maximum first derivative over all latency functions. While this is effective in avoiding oscillation effects it also slows down the routing process in a drastic way. For example, when assuming linear latency functions the obtained bounds on the convergence time depend

in a pseudopolynomial way on the ratio between the largest and the smallest coefficient over all latency functions. Remarkably, similar techniques to ensure convergence are used in well established heuristics for convex optimization. For example, Bertsekas and Tsitsiklis [2] describe a distributed algorithm for non-linear multicommodity flow in which the amount of flow that is moved in a step from one path to another depends in a linear way on the reciprocal of the second derivative of the latency functions. This algorithm can also be applied to compute Nash equilibria in the Wardrop model in a distributed way in which case the slowdown is again linear in the first derivative. An alternative approach for computing Nash equilibria based on methods from *online learning* is presented by Blum et al. [3]. Interestingly, their upper bounds on the convergence time depend in a polynomial way on the first derivative of the latency functions as well.
In this work, we show that the first derivative of the latency functions is not the limiting factor in the speed of convergence towards Nash equilibrium. We will provide upper and lower bounds that identify the "relative slope" instead of the first derivative as the relevant parameter that determines the convergence time of adaptive routing policies. This parameter is a generalization of the polynomial degree of a function. Our approach enables us to obtain the first polynomial bounds on the convergence time of adaptive rerouting policies for classes of latency functions with bounded relative slope, especially for latency functions defined by positive polynomials. Remarkably, some of our upper bounds are completely independent of any parameter reflecting the size or the structure of the network but depend only on a parameter describing the behavior of the latency functions, that is, they depend in a linear fashion on the maximum relative slope over all latency functions. A more detailed presentation of our analysis can be found in [9].

References

[1] Baruch Awerbuch, Yossi Azar, and Amir Epstein. The price of routing unsplittable flow. In *Proc. 37th Ann. ACM. Symp. on Theory of Comput. (STOC)*, 2005.

[2] Dimitri P. Bertsekas and John N. Tistiklis. *Parallel and Distributed Computing: Numerical Methods.* Athena Scientific, 1989.

[3] Avrim Blum, Eyal Even-Dar, and Katrina Ligett. Routing without regret, 2005. in preparation.

[4] George Christodoulou and Elias Koutsoupias. The price of anarchy of finite congestion games. In *Proc. 37th Ann. ACM. Symp. on Theory of Comput. (STOC)*, 2005.

[5] Richard Cole, Yevgeniy Dodis, and Tim Roughgarden. Pricing network edges for heterogeneous selfish users. In *Proc. 35th Ann. ACM. Symp. on Theory of Comput. (STOC)*, pages 521–530, 2003.

[6] Artur Czumaj and Berthold Vöcking. Tight bounds for worst-case equilibria. In *Proc. of the 13th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA)*, pages 413–420, 2002.

[7] Simon Fischer and Berthold Vöcking. On the evolution of selfish routing. In Susanne Albers and Tomasz Radzik, editors, *Proc. 12th Ann. European Symp. on Algorithms (ESA)*, number 3221 in Lecture Notes in Comput. Sci., pages 323–334, Bergen, Norway, September 2004. Springer-Verlag.

[8] Simon Fischer and Berthold Vöcking. Adaptive routing with stale information. In Marcos Kawazoe Aguilera and James Aspnes, editors, *Proc. 24th Ann. ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing (PODC).* ACM, July 2005.

[9] Simon Fischer, Harald Räcke, and Berthold Vöcking. Fast Convergence to Wardrop Equilibria by Adaptive Sampling Methods. In *Proce. of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, 2006.

[10] Lisa Fleischer. Linear tolls suffice: New bounds and algorithms for tolls in single source networks. In *Proc. 31st Int. EATCS Coll. on Automata, Languages and Programming (ICALP)*, pages 544–554, 2004.

[11] Alexis Kaporis, Efpraxia Politopoulou, and Paul Spirakis. The price of optimum in Stackelberg games. Technical report, Electronic Colloquium on Computational Complexity (ECCC), 2005.

[12] Atul Khanna and John A. Zinky. The revised ARPANET routing metric. In *Proc. ACM SIGCOMM*, pages 45–56, September 1998.

[13] Elias Koutsoupias, Marios Mavronicolas, and Paul G. Spirakis. Approximate equilibria and ball fusion. *Theory Comput. Syst.*, 36(6):683–693, 2003.

[14] James F. Kurose and Keith W. Ross. *Computer Networking, a top down approach featuring the Internet, 3rd ed.* Addison-Wesley Longman, 2004.

[15] Jennifer Rexford. *Handbook of Optimization in Telecommunications*, chapter Route optimization in IP networks. Kluwer Academic Publishers, 2005.

[16] Tim Roughgarden. Stackelberg scheduling strategies. In *Proc. 33rd Ann. ACM. Symp. on Theory of Comput. (STOC)*, pages 104–113, 2001.

[17] Tim Roughgarden. How unfair is optimal routing? In *Proc. 13th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA)*, pages 203–204, 2002.

[18] Tim Roughgarden and Éva Tardos. How bad is selfish routing? *J. ACM*, 49(2):236–259, 2002.

[19] John Glen Wardrop. Some theoretical aspects of road traffic research. In *Proc. of the Institute of Civil Engineers, Pt. II*, pages 325–378, 1952.

Arbitrarily vertex decomposable graphs

Mariusz Woźniak

Let $G = (V, E)$ be a graph of order v. A sequence (a_i) of positive integers is called *admissible* if

$$\sum_i a_i = v.$$

Let W be a family (hence a property) of graphs. We say that an admissible sequence (a_i) is *realizable* by graphs from W, if there exists a partition of the vertex set V into subsets (A_i) such that for each $i = 1, \ldots, k$

- $|A_i| = a_i$,
- the induced subgraph $G[A_i]$ belongs to W.

A graph G is called *k-decomposable with respect to W* if every admissible sequence of k integers is realizable (by graphs from W). Naturally, the problem is nontrivial if $k \geq 2$.

A graph G is *arbitrarily vertex decomposable (AVD) with respect to W* if every admissible sequence is realizable (by graphs from W).

The subject of our interest is the class of connected graphs, although some results concerning other properties are also known. ¿From now on, we assume that the property W means connectivity of graphs, and we will omit the phrase "with respect to W".

It is worth noticing that investigating arbitrarily decomposable graphs is closely related to constructions of computer networks. Namely, the problem of partitioning of the vertex set of a graph corresponds to the problem of partitioning of a large network into smaller parts used by some groups of users that can easily communicate one with another.
First results concerning k-decomposability were obtained by E. Győri [4] and, independently, by L. Lovász [8].

Theorem 1. *Every k-connected graph is k-decomposable into connected parts.*

AVD trees were investigated, at the beginning independently, by a group of French computer scientists (cf.[1]), and by M.Horňák and M.Woźniak, who conjectured in [6] that the maximal degree of every AVD tree T is at most four. This conjecture was proved by D.Barth and H.Fournier [2].
Consequently, studying AVD trees has concentrated on relatively simple structures (e.g. caterpillars) making attempts to characterize AVD ones among them (cf. [5], [3]). Recently, some other simple structures, like unicyclic graphs, were studied. (cf. [7])
In the general case, every traceable graph (i.e. having a Hamilton path) is evidently AVD. Recently, the following Ore-type theorem has been obtained by A. Marczyk [9].

Theorem 2. *If G is a two-connected graph on n vertices with the independence number at most $\lceil n/2 \rceil$ and such that the degree sum of any pair of nonadjacent vertices is at least $n-3$, then G is arbitrarily vertex decomposable.*

It would be interesting to find other sufficient conditions for graphs to be AVD which do not imply traceability. Hitherto attempts failed; these two properties occurred close each to other. Anyway, investigating relations between them seems interesting. Also the problem of k-decomposability seems interesting.
Among a couple of algorithmic aspects of the above-mentioned problems, let us mention finally the *on-line* versions of them, if they have sense. For instance, the on-line version of the problem of partitioning of a tree into subtrees can be described as follows. An admissible sequence (a_i) is not given in whole at once, but we are acquainted with it element by element. At each step we have to fix a set A_i corresponding to the element a_i, and cannot change this set later. It occurred that, as opposed to the ordinary version of the problem, a full characterization of on-line AVD trees is possible (a paper of M. Horňák, Zs. Tuza and M. Woźniak, submitted). Let's observe that the on-line version is also sensible for the problems of k-decomposability starting with $k = 3$.

References

[1] D. Barth, O. Baudon, J. Puech, *Network sharing: a polynomial algorithm for tripodes*, Discrete Applied Mathematics, **119** (2002).

[2] D. Barth, H. Fournier, *A Degree Bound on Decomposable Trees*, Preprint (2004).

[3] S. Cichacz, A. Görlich, A. Marczyk, J. Przybyło, M. Woźniak, *Arbitrarily vertex decomposable caterpillars with four or five leaves*, Preprint MD 010 (2005), www.ii.uj.edu.pl/preMD/.

[4] E. Győri, *On division of graphs to connected subgraphs*, Colloq. Math. Soc. János Bolyai, **18** (1978), 485-494.
[5] M. Horňák, M. Woźniak, *On arbitrarily vertex decomposable trees*, Preprint MD 001 (2004), www.ii.uj.edu.pl/preMD/.
[6] M. Horňák, M. Woźniak, *Arbitrarily vertex decomposable trees are of maximum degree at most six*, Opuscula Mathematica **23** (2003), 49-62.
[7] R. Kalinowski, M. Pilśniak, M. Woźniak, I.A. Zioło, *Arbitrarily vertex decomposable suns with few rays*, Preprint MD 015 (2005), www.ii.uj.edu.pl/preMD/.
[8] L. Lovász, *A homology theory for spanning trees of a graph*, Acta Math. Acad. Sci. Hungar. **30** (1977), 241-251.
[9] A. Marczyk, *A note on arbitrarily vertex decomposable graph*, Preprint MD 018 (2005), www.ii.uj.edu.pl/preMD/.

Some notes on $L(d,s)-$list labellings of trees and cacti

Anja Kohl

A special type of *Frequency Assignment Problem* (for more information we refer to [7]) are $T-$*colorings*. These are vertex colorings such that the absolute value of the difference between any two colors assigned to adjacent vertices does not belong to a prescribed set T of nonnegative integers including 0. An often studied case are $T-$colorings where T is a so-called $r-$initial set, i.e. $T = \{0, 1, \dots, r\}$.
The *distance constrained labellings* of graphs are a variation of that model. These labellings consider not only adjacent vertices but all pairs of vertices with a certain distance to each other. An often studied type of such labellings are the $L(2,1)-$labellings that first arose in 1992. The more general concept of $L(d,k)-$labellings $-$ which is the non-list version of $L(d,s)-$list labellings $-$ was introduced by Georges and Mauro [5] and was further investigated during the last decade. A good survey on this problem is [2].
Tesman [10] introduced the list version of $T-$colorings already in 1993. So it was a natural development to consider the list version of $L(d,k)-$Labellings as well. The first steps in this direction were made among others by Fiala et al. [3],[4] and Kohl et al. [8].
Throughout this abstract let $G = (V, E)$ be a simple graph with maximum degree Δ, and for all $v \in V$ let $L(v)$ be a set of labels assigned to v. This $L(v)$ is called the list of v, and the set of all lists is called the list assignment $\mathcal{L}$. A k-assignment is a list assignment where all lists have the same cardinality k, that is $|L(v)| = k$ for all $v \in V$. We assume that the labels are natural numbers.

Definition 1. *For a given list assignment $\mathcal{L} = \{L(v) \mid v \in V\}$ and nonnegative integers $d, s, (d \geq s)$ an $L(d,s)-$*list labelling *of G is a function f that assigns a label to every vertex of G, such that:*

1. $\forall v \in V : \ f(v) \in L(v)$
2. $|f(v) - f(w)| \geq d$, if $dist(v,w) = 1$
3. $|f(v) - f(w)| \geq s$, if $dist(v,w) = 2$

$\chi_\ell^{d,s}(G)$ is the smallest integer k, such that every k-assignment admits an $L(d,s)-$list labelling of G.

First let us consider $L(d,0)-$list labellings. It is easy to see that $\chi_\ell^{d,0}(G) \leq d\Delta+1$ since we can label G from lists of that length by a greedy algorithm, using in each step the smallest label occurring in the union of the lists of uncolored vertices. This bound is obviously sharp for complete graphs and odd cycles. Waller [11] proved a Brooks-type theorem stating $\chi_\ell^{d,0}(G) \leq d\Delta$ for all connected graphs G distinct from a complete graph and an odd cycle. Tesman [10] determined $\chi_\ell^{d,0}$ for all trees T_n with n vertices and he also gave in another paper the exact value of $\chi_\ell^{d,0}$ for odd cycles. Alon and Zaks [1] presented a lower bound for even cycles, which was shown to be sharp by Sitters [9]. For a path P_n and a cycle C_n with n vertices the formula is

$$(1)\quad \chi_\ell^{d,0}(T_n) = \left\lfloor \tfrac{2d(n-1)}{n} \right\rfloor + 1, \qquad \chi_\ell^{d,0}(C_n) = \begin{cases} 2d+1 & ,\text{if } 2 \nmid n \\ \left\lfloor 2d(1-\tfrac{1}{2n-1}) \right\rfloor + 1 & ,\text{if } 2 \mid n \end{cases}$$

The result for trees can also be obtained as a corollary of the following theorem:

Theorem 2. *Let T_n be a tree with n vertices. Moreover, define $\sigma^d(T_n)$ to be the smallest natural number, such that there is an $L(d,0)-$list labelling of T_n for every list assignment $\mathcal{L} = \{L(v) \mid v \in V(T_n)\}$ with the properties*

$$\sum_{v \in V(T_n)} |L(v)| \geq \sigma^d(T_n) \quad \text{and} \quad \forall v \in V(T_n): \ |L(v)| \leq 2d.$$

Then it holds $\sigma^d(T_n) = 2d(n-1)+1$.

Next let us consider cacti. A cactus is a finite, connected graph such that every edge belongs to at most one cycle. For cacti containing at least one cycle we have:

Theorem 3. *Let C be a cactus of order n and girth $g \geq 3$. If C contains exactly one cycle, then $\chi_\ell^{d,0}(C) = 2d+1$ if g is odd and $2d+1-\min\{\lceil \frac{2d}{n} \rceil, \lceil \frac{2d}{2g-1} \rceil\} \leq \chi_\ell^{d,0}(C) \leq 2d$ if g is even. If C has at least two cycles, then $\chi_\ell^{d,0}(C) \leq \lceil \frac{2dg-1}{g-1} \rceil$.*

For $L(d,s)-$list labellings where $s > 0$ only a few results are known so far, e.g. the exact value of $\chi_\ell^{2,1}$ for paths and cycles given by Fiala and Škrekovski [3]:

$$(2)\qquad \chi_\ell^{2,1}(P_n) = \begin{cases} 3 & ,\text{if } n = 2 \\ 4 & ,\text{if } n = 3,4 \\ 5 & ,\text{if } n \geq 5 \end{cases}, \qquad \chi_\ell^{2,1}(C_n) = 5.$$

Moreover, for stars it holds $\chi_\ell^{1,1}(K_{1,n}) = n+1$ and if $d \geq 2$:

(i) $\chi_\ell^{d,s}(K_{1,n}) \leq 2d-1+s(n-1)$, (Voigt [8])

(ii) $\chi_\ell^{d,s}(K_{1,n}) \geq \lfloor \frac{2dn}{n+1} \rfloor + 1$, (Tesman [10])

(iii) $\chi_\ell^{d,s}(K_{1,n}) \geq d+s(n-1)+1$, (Georges and Mauro [5])

(iv) $\chi_\ell^{d,1}(K_{1,n}) \geq \lceil (2d-1)\frac{a}{a+1} \rceil + n - 1$ for $a := \lfloor \frac{n}{2d} \rfloor$. (Tuza [8])

Theorem 4. *If $n \geq \lceil \frac{2a}{(d-a-1)} \rceil (d+a)+1$ for an $a \in \{0,1,\ldots,d-2\}$, then $\chi_\ell^{d,1}(K_{1,n}) \geq d+a+n$.*

Theorem 5. *If $d \geq s$, then $\chi_\ell^{d,s}(K_{1,n}) \leq \left\lfloor \frac{[2d+s(n-1)]n}{n+1} \right\rfloor + 1$.*

For paths we have $\lfloor \frac{2d(n-1)}{n} \rfloor + 1 \leq \chi_\ell^{d,1}(P_n) \leq \lfloor \frac{2d(n-1)}{n} \rfloor + 3$ (Voigt [8]). Furthermore, it holds $\chi_\ell^{d,s}(P_n) \leq 2d + 2s - 1$ that can be achieved by a greedy algorithm. By constructing specific list assignments we can further prove

Theorem 6. *For a path P_n, $n \geq 3$ it holds $\chi_\ell^{d,d}(P_n) > \lfloor 3d \left(1 - \frac{1}{n}\right) \rfloor$.*

Our intuition says that this lower bound is already best possible, so we pose the following conjecture:

Conjecture 2. $\chi_\ell^{d,d}(P_n) = \lfloor 3d \left(1 - \frac{1}{n}\right) \rfloor + 1$.

A second conjecture that is related to the previous one is

Conjecture 3. $\chi_\ell^{d,s}(P_n) \leq \lfloor (2d+s) \left(1 - \frac{1}{n}\right) \rfloor + 1$.

For all trees T and $d, s \geq 1$ it holds $\chi_\ell^{d,s}(T) \leq s\Delta + 2d - 1$, proved by Voigt [8]. Thus, $\chi_\ell^{2,1}(T) \leq \Delta + 3$. For the non-list version of $L(2,1)-$list labellings that can be obtained by using the list assignment that assigns the list $\{1, 2, \ldots, k\}$ to every vertex $v \in V(T)$ we already know that $k =: \chi^{2,1}(T) \geq \Delta + 2$ (Griggs and Yeh [6]).

Conjecture 4. *For every tree T it holds $\chi_\ell^{2,1}(T) = \chi^{2,1}(T)$.*

This conjecture is obviously true for stars and paths, but we could also verify it for comets (i.e. trees of radius 2) and specific caterpillars.
For cacti we investigated $L(d,1)-$list labellings and obtained the results:

Theorem 7. *If $\Delta = 3$ and the cactus C contains a cycle of length 5, then $\chi_\ell^{1,1}(C) = 5$. Otherwise $\chi_\ell^{1,1}(C) = \Delta + 1$ for $\Delta \geq 3$.*

Theorem 8. *Let C be a cactus with maximum degree $\Delta \geq 3$, girth $g \geq 3$ and $d \geq 2$. Define $a := \left\lfloor \frac{2d-2}{g-3} \right\rfloor$. Then*

$$\chi_\ell^{d,1}(C) \leq \begin{cases} \Delta + 3d - 2 & , \text{ if } g = 3 \\ \Delta + 2d - 1 & , \text{ if } g = 4 \wedge \Delta \geq 2d + 1 \\ \lfloor \frac{8d+2\Delta}{3} \rfloor & , \text{ if } g = 4 \wedge \Delta \leq 2d \\ \Delta + 2d - 1 & , \text{ if } g \geq 5 \wedge \Delta \geq 4 + a \\ \Delta + 2d & , \text{ if } g \geq 5 \wedge \Delta = 3 + a \\ \left\lfloor \frac{2dg+2\Delta-6}{g-1} \right\rfloor + 3 & , \text{ if } g \geq 5 \wedge \Delta \leq 2 + a \end{cases} .$$

References

[1] N. Alon, A. Zaks, T-choosability in graphs, Discrete Applied Math. 82 (1998), 1–13.

[2] T. Calamoneri, The $L(h,k)-$Labelling Problem: a Survey, Tech. Rep. 04/2004, Dept. of Comp. Sci. Univ. of Rome ”La Sapienza”.

[3] J. Fiala, R. Škrekovski, List distance-labelings of graphs, Discrete Applied Math. 148 (1) (2005), 13-25.

[4] J. Fiala, D. Král, R. Škrekovski, A Brooks-type Theorem for the Generalized List $T-$Coloring, SIAM Journal on Discrete Mathematics 19 (3) (2005), 588-609.

[5] J.P. Georges, D.W. Mauro, Generalized Vertex Labelings with a Condition at Distance Two, Congressus Numerantium 109 (1995), 141-159.

[6] J.R. Griggs, R.K. Yeh, *Labelling Graphs with a Condition at Distance 2*, SIAM J. Disc. Math. 5 (4) (1992), 586-595.

[7] W.K. Hale, *Frequency assignment: theory and applications*, Proc. IEEE 68 (1980), 1497-1514.

[8] A. Kohl, J. Schreyer, Zs. Tuza, M. Voigt, List version of $L(d,s)-$labelings, Theoretical Computer Science 349 (1) (2005), 92-98.

[9] R.A. Sitters, A short proof of a conjecture on the T_r-choice number of even cycles, Discrete Applied Math. 92 (1999), 243-246.

[10] B.A. Tesman, List $T-$colorings of graphs, Discrete Applied Math. 45 (1993), 277-289.

[11] A.O. Waller, *Some results on list $T-$colourings*, Discrete Math. 174 (1997), 357-363.

On Sparse Normal Graphs

Annegret K. Wagler
(joint work with B. Randerath)

A graph G is called *normal* if G admits a clique cover $\mathcal{Q}$ and a stable set cover $\mathcal{S}$ such that every clique in $\mathcal{Q}$ intersects every stable set in $\mathcal{S}$.
Figure 1 presents two normal graphs (the bold edges are the clique covers and $\{\{1,3,5\},\{1,4,6\},\{2,4,5,7\}\}$ resp. $\{\{0,2,4,6\}, \{0,3,5,7\}, \{1,3,6\}\}$ the cross-intersecting stable set covers).

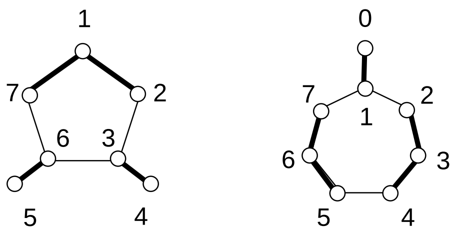


Figure 1. Two normal graphs

The interest in normal graphs is caused by the fact that they form a weaker variant of the well-known perfect graphs by, e.g., means of co-normal products [3] and graph entropy [2]. Perfect graphs have been recently characterized as those graphs without odd holes C_{2k+1} and odd antiholes $\overline{C}_{2k+1}$ as induced subgraphs (Strong Perfect Graph Theorem [1]). In analogy, Körner and de Simone [4] conjectured that every $(C_5, C_7, \overline{C}_7)$-free graph is normal (Normal Graph Conjecture).
The Normal Graph Conjecture is asymptotically true since already almost all C_5-free graphs are perfect according to Prömel and Steger [5] who proved that graphs with middle edge densities almost surely contain a C_5 as induced subgraph. Therefore, a graph is with high probability perfect only if it is too sparse or, due to the invariance of perfectness by complementation, too dense to contain a C_5.

As this result could also imply that there are not many more normal than perfect graphs, we shall relate the two classes, starting by studying *sparse* graphs. Such graphs typically consist of many small *1-tree* components, that are connected graphs with as many edges as nodes. Thus, we start with 1-trees. Moreover, as adding random edges to such sparse graphs links different 1-trees to larger components, we extend our study to so-called cacti, obtained by linking 1-trees together.
A 1-tree can be obtained from a (chordless) cycle and certain trees by a sequence of node-identifications. Since all trees and all cycles different from C_5 and C_7 are normal and node-identification preserves normality [8], this already implies:

Corollary 1. *The Normal Graph Conjecture is true for 1-trees and their complements.*

Let $G_1 +_v G_2$ denote the graph obtained from G_1 and G_2 by identification in the node v. We fully characterize the normal 1-trees as follows:

Theorem 2. *A 1-tree G is* not *normal iff one of the following holds.*
(i) $G = C_5$.
(ii) $G = C_5 +_v T$ *where T is a tree.*
(iii) $G = (C_5 +_v T) +_{v'} T'$ *where T, T' are trees and v, v' are two nodes of the C_5 at distance two.*
(iv) $G = C_7$.

This implies in particular that there are many more normal than perfect 1-trees, since a 1-tree is perfect if and only if its only cycle is even or a triangle, whereas almost all 1-trees are normal.
We extend this result further to the larger class of cacti. A *cactus* is a connected graph whose cycles are all edge-disjoint. Thus, a cactus $G = (V, E)$ with k cycles can be considered as a graph obtained from a tree by adding k edges in a certain way (thus cacti admit $|V| - 1 + k$ edges and are still sparse). Alternatively, every cactus can be obtained from several 1-trees by a sequence of node-identifications. Thus, we can apply our characterization of the normal 1-trees and our knowledge on the behavior of normal graphs under node-identification from [8] in order to figure out which cacti are normal. Since all (C_5, C_7)-free 1-trees are in particular normal by Theorem 2 and node-identification preserves normality [8], we can already infer:

Corollary 3. *The Normal Graph Conjecture is true for cacti and their complements.*

Obviously, there also exist normal cacti containing a C_5 or a C_7 as induced subgraph, for instance the graph obtained by identifying the two graphs from Figure 1 in a node.
We develop an algorithm that decides in polynomial time whether a given cactus G is normal: As a first step, we decompose G accordingly into as many 1-trees as G has cycles by choosing an appropriate set of cut-nodes in G. As a second step, we decide with the help of Theorem 2 whether the resulting 1-tree components of

G are normal or not. As main step, we succesively identify two such components of G in their common node and decide whether this yields a normal graph (note that we can obtain a normal graph $G_1 +_v G_2$ by node-identification if G_1 is normal but G_2 not [8], for instance the normal graphs in Figure 1 are obtained by identifying a non-normal graph and an edge in a node).
Our results imply that almost all sparse graphs are normal; we conclude that there are many more normal than perfect sparse graphs. First results on classes of denser graphs verify the Normal Graph Conjecture for circulants [7] and line graphs [6] and show that they contain more normal than perfect graphs as well. However, it is open with which probability random graphs with middle edge density are normal.

References

[1] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas, *The Strong Perfect Graph Theorem*, to appear in Annals of Mathematics.
[2] I. Cziszár, J. Körner, L. Lovász, K. Marton, and G. Simonyi. *Entropy splitting for antiblocking corners and perfect graphs*, Combinatorica 10 (1990) 27–40.
[3] J. Körner, *An Extension of the Class of Perfect Graphs*, Studia Math. Hung. 8 (1973) 405–409.
[4] J. Körner and C. de Simone, *On the Odd Cycles of Normal Graphs*, Discrete Appl. Math. 94 (1999) 161–169.
[5] H.J. Prömel and A. Steger, *Almost all Berge graphs are perfect*, Combinatorics, Probability, and Computing 1 (1990) 53–79.
[6] H.O. Schülzke, *The Normal Graph Conjecture for line graphs.* Diplomarbeit, TU Berlin, 2006.
[7] A. Wagler, *The Normal Graph Conjecture is true for circulants*, to appear in: Proceedings of GT04, Birkhäuser.
[8] A. Wagler, *Constructions for Normal Graphs and some Consequences*, submitted to Discrete Applied Mathematics.

Distributed Coloring in $\tilde{O}(\sqrt{\log n})$ Bit Rounds

Christian Scheideler
(joint work with Kishore Kothapalli, Melih Onus, Andrea Richa)

We consider the well-known vertex coloring problem: given a graph G, find a coloring of the vertices so that no two neighbors in G have the same color. It is trivial to see that every graph of maximum degree Δ can be colored with $\Delta + 1$ colors, and distributed algorithms that find a $(\Delta + 1)$-coloring in a logarithmic number of communication rounds, with high probability, are known since more than a decade. This is in general the best possible if only a constant number of bits can be sent along every edge in each round. In fact, we show that for the n-node cycle the *bit complexity* of the coloring problem is $\Omega(\log n)$. More precisely, if only one bit can be sent along each edge in a round, then *every* distributed coloring algorithm (i.e., algorithms in which every node has the same initial state and initially only knows its own edges) needs at least $\Omega(\log n)$ rounds, with high probability, to color the cycle, for *any* finite number of colors. But what if the

edges have orientations, i.e., the endpoints of an edge agree on its orientation (while bits may still flow in both directions)? Edge orientations naturally occur in dynamic networks where new nodes establish connections to old nodes. Does this allow one to provide faster coloring algorithms?
Interestingly, for the cycle in which all edges have the same orientation, we show that a simple randomized algorithm can achieve a 3-coloring with only $O(\sqrt{\log n})$ rounds of bit transmissions, with high probability (w.h.p.). This result is tight because we also show that the bit complexity of coloring an oriented cycle is $\Omega(\sqrt{\log n})$, with high probability, no matter how many colors are allowed. The 3-coloring algorithm can be easily extended to provide a $(\Delta + 1)$-coloring for all graphs of maximum degree Δ in $O(\sqrt{\log n})$ rounds of bit transmissions, w.h.p., if Δ is a constant, the edges are oriented, and the graph does not contain an oriented cycle of length less than $\sqrt{\log n}$. Using more complex algorithms, we show how to obtain an $O(\Delta)$-coloring for arbitrary oriented graphs of maximum degree Δ using essentially $O(\log \Delta + \sqrt{\log n})$ rounds of bit transmissions, w.h.p., provided that the graph does not contain an oriented cycle of length less than $\sqrt{\log n}$.
In this abstract, we only give details about selected aspects of our work. More details and related work can be found in [1]. In Section 1 we define our model, in Section 2 we present the lower bound and in Section 3 we present the upper bound for distributed coloring in oriented graphs.

1. Model and Definitions

We model the distributed system as a graph $G = (V, E)$ with V representing the set of computing entities, or processors, and $E \subseteq V \times V$ representing all the available communication links. We assume that all the communication links are undirected and hence bidirectional. All the processors start at the same time and time proceeds in synchronized rounds. We let $n = |V|$. The degree of node u is denoted d_u and by Δ we denote the maximum degree of G, i.e., $\Delta = \max_{u \in V} d_u$. When there is no confusion, d_u will also be used to refer to the number of uncolored neighbors of node u. By N_u we denote the set of neighbors of node u and when there is no confusion, we use N_u to refer to the set of uncolored neighbors of u. We do not require that the nodes in V have unique labels of any kind. For our algorithms to work, it is enough that each node knows a constant factor estimate of the logarithm of the size of the network apart from its own degree and neighbors. When we consider graphs of constant degree, *no* global knowledge is required for our algorithm and it suffices that each node knows its own degree.
Let us denote by $[x]$ the set $\{1, 2, \ldots, x\}$ if $x \in \mathbb{N}$. If $x \in \mathbb{R}^+$, then $[x]$ would be the set $\{1, 2, \ldots, \lceil x \rceil\}$. Given a graph $G = (V, E)$ a vertex coloring is a mapping $c : V \to [C]$ such that if $\{u, v\} \in E$ then $c(u) \neq c(v)$, i.e., no two adjacent vertices receive the same color. Here C denotes the number of colors used in the coloring. We say that a coloring is a *local coloring* if every node u with degree d_u has a color in $[\epsilon d_u]$ when the coloring uses $\epsilon\Delta$ colors.
In our model, the measure of efficiency is the number of bits exchanged. We also refer to this as the *bit complexity*. We view each round of the algorithm as

consisting of 1 or more *bit rounds*. In each bit round each node can send/receive at most 1 bit from each of its neighbors. We assume that the rounds of the algorithm are synchronized. The bit complexity of algorithm A is then defined as the number of bit rounds required by algorithm A. We note that, since the nodes are synchronized, each round of the algorithm requires as many bit rounds as the maximum number of bit rounds needed by any node in this round. In our model, we do not count local computation performed by the nodes. This is reasonable as in our algorithms nodes perform only simple local computation.
In our model, we assume that the edges in E have an orientation associated with them. That is, for any two neighbors v, w exactly one of the following holds for the edge $\{v, w\}$: $\{v, w\}$ is oriented either $v \to w$ or as $w \to v$. In the former we also call v *superior* to w and vice-versa in the latter. Having orientation on the edges is a property that has not been studied in the context of vertex coloring though it is a natural property since networks usually evolve and for every connection there is usually a node that initiated it. We show that algorithms for symmetry breaking can be greatly improved provided that the underlying graph is oriented. The exact way in which orientation is used for symmetry breaking is explained in Figure **??**. As shown, if nodes v and w choose the same color during any round of the algorithm, in the existing algorithms, both nodes remain uncolored as in Figure **??**(b) and have to try in a later round. With orientation, if the edge $\{v, w\}$ is oriented as $v \to w$, then node v can retain its choice provided that there is no edge $\{u, v\}$ oriented $u \to v$ and u also chooses the same color.
One parameter that will be important for our investigations is the length of the shortest cycle in the orientation. We formalize this notion in the following definition.

Definition 1 *(ℓ–acyclic Orientation)*. *An orientation of the edges of a graph is said to be ℓ–acyclic if the minimum length of any directed cycle induced by the orientation is at least ℓ. Note that this is not the girth of the given graph.*

2. The lower bound

In this section we establish lower bounds on the bit complexity of finding a proper vertex coloring. Recall that a Las Vegas algorithm is a randomized algorithm that always produces a correct result, with the only variation being its runtime. Notice that the lower bound holds for any finite number of colors.

Theorem 2. *For every Las Vegas algorithm A there is an infinite family of oriented graphs s.t. A has a bit complexity of at least $\Omega(\sqrt{\log n})$ on this family, with high probability, to compute a proper vertex coloring.*

Proof. Consider the cycle of n nodes in which all the edges are oriented in the same direction. Let $S_\ell = (u_\ell, \dots, u_1, v_1, \dots, v_\ell)$ be the set of nodes along a path of length 2ℓ of the cycle. Initially, every node in S_ℓ is in the same state s_0. Associated with s_0 is a fixed probability distribution $P_0 = (p^0_{x,y})_{x,y\in\{-,0,1\}}$ for sending bit x along the left edge and bit y along the right edge, where "$-$" represents the case that no bit is sent. Since P_0 has only nine probability values, there must be an

x_0 and y_0 with $p^0_{x_0,y_0} \geq 1/9$. Let E_1 be the event that all nodes in S_ℓ choose that option. Then all nodes in $S_{\ell-1} = (u_{\ell-1}, \ldots, u_1, v_1, \ldots, v_{\ell-1})$ receive the same information and must therefore be in the same state s_1. Associated with s_1 is a fixed probability distribution $P_1 = (p^1_{x,y})_{x,y\in\{-,0,1\}}$ for sending bit x along the left edge and bit y along the right edge. Since P_1 has only nine probability values, there must be an x_1 and y_1 with $p^1_{x_1,y_1} \geq 1/9$. Let E_2 be the event that all nodes in $S_{\ell-1}$ choose that option. Then all nodes in $S_{\ell-2}$ receive the same information and must therefore be in the same state s_2.
Continuing with this argumentation, it follows that there are events $E_1, \ldots, E_\ell$ with E_i having a probability of at least $(1/9)^{2(\ell-i+1)}$ for all i so that all nodes in $S_{\ell-i}$ are in the same state s_i. Since these nodes are neighbors, algorithm A cannot terminate within ℓ bit exchanges if $E_1, \ldots, E_\ell$ are true because whatever probability distribution A chooses on the colors, the probability that two neighboring nodes choose the same color is non-zero, which would violate the assumption that A is a Las Vegas algorithm.
The probability that $E_1, \ldots, E_\ell$ are true is at least $\left(\frac{1}{9}\right)^{\sum_{i=1}^{\ell} 2(\ell-i+1)} \geq \left(\frac{1}{9}\right)^{\ell^2/2}$ and when choosing $\ell = \sqrt{2\log_9(n/2\log^2 n)}$, this results in a probability of at least $(2\log^2 n)/n$. Moreover, notice that $E_1, \ldots, E_\ell$ only depend on the nodes in S_ℓ because information can only travel a distance of ℓ edges in ℓ rounds. Hence, we can partition the n-node cycle into $n/2\ell$ many sequences S where each sequence has a probability of at least $\frac{2\log^2 n}{n}$ of running into the events $E_1, \ldots, E_\ell$ that is independent of the other sequences. Hence, the probability that all node sequences can avoid the event sequence $E_1, \ldots, E_\ell$, which is necessary for A to terminate, is at most $\left(1 - \frac{2\log^2 n}{n}\right)^{n/2\ell} \leq 1/n$, which implies that A needs $\Omega(\sqrt{\log n})$ bit-rounds, with high probability, to finish. □

3. The upper bound for constant degree graphs

In this section we present and analyze the algorithm for $(\Delta+1)$–coloring constant degree oriented graphs. The algorithm for vertex coloring constant degree oriented graphs is given in Figure 1. In the algorithm, the parameter C_u refers to the number of colors used in the coloring by node u. Each node executes the algorithm Color-Random until it gets colored.

Theorem 3. *Given a $\sqrt{\log n}$–acyclic oriented graph $G = (V, E)$ of maximum degree Δ, if Δ is a constant, a $(\Delta+1)$–vertex coloring of G can be obtained in $O(\sqrt{\log n})$ bit rounds, with high probability.*

Proof. The analysis below cuts the time into two phases. Phase I ends once every simple oriented path of length $\ell = \sqrt{\log n}$ has at least one colored node, and phase II ends once all nodes are colored. We show that phase I takes at most $r = 4\sqrt{\log n}$ rounds, with high probability. For Phase II, the proof uses the $\sqrt{\log n}$–acyclic orientation to argue that a further $\sqrt{\log n}$ rounds suffice to color all nodes. For simplicity, we set $C_u = 2\Delta$ for every node u, but the analysis works, with minor modifications, for $C_u = \Delta + 1$, as long as Δ is a constant.

```
Algorithm Color-Random($C_u$)
  While $u$ is not colored do
    1. Node $u$ chooses a color $c_u$ from the available colors in $[C_u]$
       uniformly at random.
    2. Node $u$ communicates its choice $c_u$, from step 1, to all of its
       uncolored neighbors that have a lower priority over $u$, i.e.
       to nodes $v$ such that $u \to v$.
    3. If node $u$ does not receive a message from any of its neighbors
       $w$ with $w \to u$ and $c_w = c_u$, then node $u$ gets colored with
       color $c_u$. Otherwise node $u$ remains uncolored.
    4. If $u$ is colored during step 3 of the current round, then $u$
       informs all of its uncolored neighbors about the color of $u$.
    5. Node $u$ updates the list of available colors according to colors
       taken up by $u$'s neighbors.
```

FIGURE 1. Coloring constant degree oriented graphs by random choices.

Consider any simple oriented path P of length ℓ. For any node $u \in P$ with C'_u remaining colors and d'_u remaining uncolored neighbors, the probability that it chooses a color that is identical to the choice of any of its uncolored neighbors is at most $\sum_{j=1}^{d'_u} 1/C'_u \leq d'_u/(2\Delta - (d_u - d'_u)) \leq 1/2$ as $C'_u = 2\Delta - (d_u - d'_u)$ and $d'_u \leq d_u$.
For any $i \geq 1$, denote by $E_{P,i}$ the event that all nodes in P have a color conflict in round i. Since each node chooses the color independently and uniformly at random, and P is oriented, one can identify a distinct witness for each color conflict so as to upper bound $\Pr[E_{P,i} \mid \cap_{j=0}^{i-1} E_{P,j}]$ as $\Pr[E_{P,i} \mid \cap_{j=0}^{i-1} E_{P,j}] \leq (1/2)^{\ell}$.
Denote by E_P the event that the event $E_{P,i}$ occurs for r consecutive rounds. Then,

$$\Pr[E_P] = \Pr[\bigcap_{i=1}^{r} E_{P,i}] = \Pi_{i=1}^{r} \ \Pr[E_{P,i} \mid \cap_{j=1}^{i-1} E_{P,j}] \leq (1/2)^{\ell r}.$$

Let E denote the event that for some simple oriented path P the event E_P occurs. The number of simple oriented paths of length ℓ is at most $n\Delta^{\ell}$ by choosing the first vertex from n available choices and choosing each of the next ℓ vertices from the at most Δ available choices. Thus,

$$\Pr[E] = \Pr[\bigcup_{P} E_P] \leq n\Delta^{\ell} \Pr[E_P] \leq 1/n^2.$$

for the above value of r since $\Delta = O(1)$. This completes Phase I of the analysis.
Consider connected components of uncolored nodes. At the end of Phase I, since any simple oriented path of length ℓ has at least one colored node, each such component only has simple oriented paths of length less than ℓ, with high probability. Moreover, the input graph does not have oriented cycles of length less than $\sqrt{\log n}$ which implies that each such component can be organized into less than $\sqrt{\log n}$ layers with oriented edges going only from a node in a lower-numbered layer to a

node in a higher numbered layer. This layering can be achieved by the following process. Nodes with no superiors are assigned to layer 0. After removing these nodes, nodes in the rest of the component with no superiors are assigned to layer 1, and so on, until there are no nodes left. Such a procedure terminates in less than $\sqrt{\log n}$ rounds, implying that the layer number of any node is less than $\sqrt{\log n}$. Otherwise, there must exist either a simple oriented path of length at least $\sqrt{\log n}$ or an oriented cycle of length less than $\sqrt{\log n}$. Both of these conditions result in a contradiction and hence the layering process must terminate in less than $\sqrt{\log n}$ rounds.
Now, in Phase II, during every round the uncolored nodes assigned to the lowest layer number presently get colored as the nodes assigned to the lowest layer can always retain their color choice from Step 1. This implies that Phase II can finish in less than $\sqrt{\log n}$ rounds.
Since in each round each uncolored node has to exchange $O(\log \Delta) = O(1)$ bits, the bit complexity of the algorithm Color-Random is $O(\sqrt{\log n})$. $\square$

We note that the same proof also holds for 3–coloring cycle graphs, with any orientation, with minimal changes. Coupled with the lower bound result in Theorem 2, our analysis for the case of constant degree graphs is tight with respect to the bit complexity, up to constant factors. The algorithm and the analysis can be modified easily to achieve a local coloring also.

References

[1] K. Kothapalli, M. Onus, A. Richa and C. Scheideler, *Distributed Coloring in $\tilde{O}(\sqrt{\log n})$ Bit Rounds*, In Int. Parallel and Distributed Processing Symposium (IPDPS), 2006.

Algebraic Graph Algorithm: From Shortest Paths to Matchings

Piotr Sankowski

In this paper we consider the problem of finding maximum weighted matchings in bipartite graphs with nonnegative integer weights. The presented algorithms for this problems work in $\tilde{O}(Wn^{\omega})$ time, where ω is the matrix multiplication exponent, and W is the highest edge weight in the graph. The best bound on $\omega \leq 2.376$ is due to Coppersmith and Winograd [2].
The weighted matching problem is one of the fundamental problems in combinatorial optimization. The first algorithm for this problem in the bipartite case was proposed in the fifties of the last century by Kuhn [12]. His result has been improved several times since then, the results are summarized in the Table 1.

The bold font indicates an asymptotically best bound in the tables. In particular the presented here algorithm is faster than the algorithm of Gabow and Tarjan [6] and the algorithm of Edmonds and Karp [4] in the case of dense graphs with small integer weights. Note, that in the above summary there are no algorithms that use matrix multiplication. However, in the papers studying the parallel complexity of

Complexity	Author
$O(n^4)$	Khun (1955) [12] and Munkers (1957) [16]
$O(n^2m)$	Iri (1960) [8]
$O(n^3)$	Dinic and Kronrod (1969) [3]
$\boldsymbol{O(nm)}$	Edmonds and Karp (1970) [4]
$O(n^{\frac{3}{4}}m\log W)$	Gabow (1983) [5]
$\boldsymbol{O(\sqrt{n}m\log(nW))}$	Gabow and Tarjan (1989) [6]
$O(\sqrt{n}mW)$	Kao, Lam, Sung and Ting (1999) [10]
$\boldsymbol{O(n^\omega W)}$	this paper

Table 1: The complexity results for the bipartite weighted matching problem.

the problem [11, 15], such algorithms are implicitly constructed. These results lead to $O(Wn^{\omega+2})$ sequential time algorithms. In this paper we improve the complexity by factor of n^2. The improvement in the exponent by 1 is achieved with use of the very recent results of Storjohann [19], who had shown faster algorithms for computing polynomial matrix determinants. Further improvement is achieved by a novel reduction technique, that allows us to reduce the weighted version of the problem to unweighted one. The four steps of the reduction are schematically presented on Figure 1. As a step of the reduction we also compute the bipartite weighted cover of the graph. The unweighted problem is then solved with use of the $O(n^\omega)$ time algorithms developed last year by Mucha and Sankowski [14]. Storjohann's result can also be used to compute the maximum weight of a perfect matching in general graphs. However, the problem of finding such matching remains unsolved.

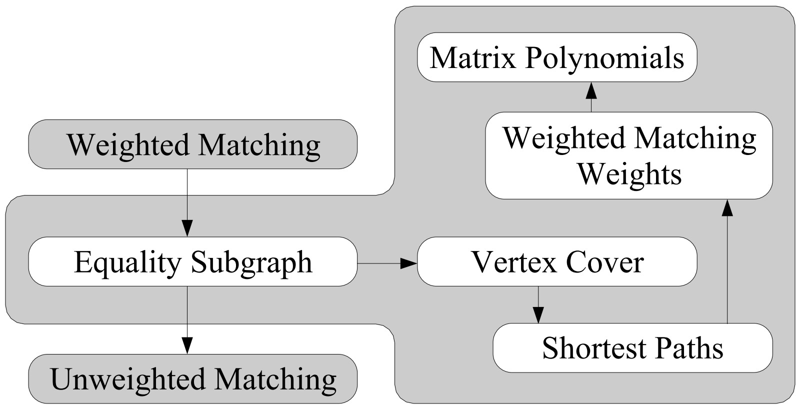


FIGURE 1. The scheme of the reduction from weighted to unweighted matchings.

The weighted matching problem is not only interesting by itself, but also it can be used to solve many other problems in combinatorial optimization. In particular, the presented algorithm for finding maximum weighted perfect matchings can be used to find minimum weighted perfect matching, as well as, maximum and minimum weighted matchings. Moreover, the minimum weighted perfect matching algorithm can be used for computing the minimum weight of k vertex disjoint s-t paths, whereas the minimum weighted vertex cover can be used to solve the single source shortest paths (SSSP) problem with negative edge weights. The complexity of the algorithms for computing the minimum weight of k vertex disjoint s-t paths, follow exactly the results in Table 1. The author is not aware of any special algorithms for this problem. The complexity results for the SSSP problem with negative edge weights are summarized in Table 2.

Complexity	Author
$O(n^4)$	Shimbel (1955) [18]
$O(n^2mW)$	Ford (1956) [9]
$\boldsymbol{O(nm)}$	Bellman (1958) [1], Moore (1959) [13]
$O(n^{\frac{3}{4}}m\log W)$	Gabow (1983) [5]
$O(\sqrt{n}m\log(nW))$	Gabow and Tarjan (1989) [6]
$\boldsymbol{O(\sqrt{n}m\log(W))}$	Goldberg (1993) [7]
$\boldsymbol{\tilde{O}(n^{\omega}W)}$	Sankowski (2005) [17], Yuster and Zwick (2005) [20], this paper

Table 2: The complexity results for the SSSP problem with negative weights. The bold font indicates an asymptotically best bound in the table.

References

[1] R. Bellman. On a Routing Problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[2] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 1–6. ACM Press, 1987.

[3] E.A. Dinic and M. A. Kronrod. An Algorithm for the Solution of the Assignment Problem. *Soviet Math. Dokl.*, 10:1324–1326, 1969.

[4] J. Edmonds and R.M. Karp. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *J. ACM*, 19(2):248–264, 1972.

[5] H.N. Gabow. Scaling Algorithms for Network Problems. *J. Comput. Syst. Sci.*, 31(2):148–168, 1985.

[6] H.N. Gabow and R.E. Tarjan. Faster Scaling Algorithms for Network Problems. *SIAM J. Comput.*, 18(5):1013–1036, 1989.

[7] Andrew V. Goldberg. Scaling algorithms for the shortest paths problem. In *SODA '93: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 222–231. Society for Industrial and Applied Mathematics, 1993.

[8] M. Iri. A new method for solving transportation-network problems. *Journal of the Operations Research Society of Japan*, 3:27–87, 1960.

[9] L.R. Ford Jr. Network Flow Theory. Paper P-923, The RAND Corperation, Santa Moncia, California, August 1956.

[10] M.-Y. Kao, T. W. Lam, W.-K. Sung, and H.-F. Ting. A decomposition theorem for maximum weight bipartite matchings with applications to evolutionary trees. In *Proceedings of the 7th Annual European Symposium on Algorithms*, pages 438–449, 1999.
[11] R. M. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is in random nc. *Combinatorica*, 6(1):35–48, 1986.
[12] H.W Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.
[13] E. F. Moore. The Shortest Path Through a Maze. In *Proceedings of the International Symposium on the Theory of Switching*, pages 285–292. Harvard University Press, 1959.
[14] Marcin Mucha and Piotr Sankowski. Maximum matchings via gaussian elimination. In *Proceedings of the 45th annual IEEE Symposium on Foundations of Computer Science*, pages 248–255, 2004.
[15] K. Mulmuley, U.V. Vazirani, and V.V. Vazirani. Matching is as easy as matrix inversion. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 345–354. ACM Press, 1987.
[16] J. Munkres. Algorithms for the Assignment and Transportation Problems. *Journal of SIAM*, 5(1):32–38, 1957.
[17] P. Sankowski. Shortes paths in matrix multiplication time. In *Proceedings of the 13th Annual European Symposium on Algorithms, LNCS 3669*, pages 770–778, 2005.
[18] A. Shimbel. Structure in Communication Nets. In *In Proceedings of the Symposium on Information Networks*, pages 199–203. Polytechnic Press of the Polytechnic Institute of Brooklyn, Brooklyn, 1955.
[19] Arne Storjohann. High-order lifting and integrality certification. *J. Symb. Comput.*, 36(3-4):613–648, 2003.
[20] R. Yuster and U. Zwick. Answering distance queries in directed graphs using fast matrix multiplication. In *In Proceedings of the 46th Annual Symposium on Foundations of Computer Science*, pages 90–100. IEEE, 2005.

On Relaxation for the Maximum Acyclic Subgraph Problem

Alantha Newman

Given a directed, weighted graph $G = (V, A)$, the *maximum acyclic subgraph* problem is to find a maximum weight subset of the edges that is acyclic. If we require that all edge weights are non-negative, then this problem is equivalent to the *linear ordering problem*, which is to find a linear ordering of the vertices that maximizes the weight of the forward edges. And edge (i, j) is *forward* with respect to a vertex ordering if i precedes vertex j in the ordering. The max acyclic subgraph problem is NP-hard to approximate to within better than 65/66 [7]. It has a simple $\frac{1}{2}$-approximation: take an arbitrary ordering of the vertices. Either the set of forward edges or the set of backwards edges contains at least half of the total edge weight and each set is acyclic. Improving upon the factor of $\frac{1}{2}$ is a challenging open problem.

The aforementioned approximation algorithm uses a trivial upper bound of the total edge weight. In order to improve upon the factor of $\frac{1}{2}$, we first need to be able to compute an accurate upper bound for graphs, for example, in which the maximum acyclic subgraph has weight close to half the total edge weight. Classical linear programming relaxations, based on cycle constraints, can give bounds on the optimal value of a solution that are as much as a factor of 2 larger than the

value of an actual optimal solution. In other words, there exists infinitely many graphs which have a maximum acyclic subgraph close to half the edges, but for which the classical linear programming relaxations yield objective values close to the all the edges.
Semidefinite programming relaxations were first used in approximation algorithms by Goemans and Williamson [1]. They applied it to the maximum cut problem to obtain algorithms that yielded a much better approximation ratio than the ratio obtainable via known linear programming relaxations. For many years, it was an open problem to obtain a better-than-half approximation for the maximum cut problem. The work of Delorme, Poljak and Rendl provided hope that such an approximation could be achieved by demonstrating experimentally that the so-called eigenvalue bound yielded more accurate upper bounds on the value of an optimal cut for classes of graphs on which the classical linear programming relaxations performed poorly [2, 3, 4].
We discuss semidefinite programming formulations for the maximum acyclic subgraph problem. These formulations are based on using semidefinite constraints to model assignment constraints, whose integral solutions represent permutations. In other words, the SDP formulation we consider is a relaxation of a quadratic integer program whose feasible solutions are all permutations of the vertices. This is in contrast to the classical linear programming formulations, which are relaxations of integer programs whose feasible solutions are all acyclic subsets of the edges.
Our main theorem is that these SDP relaxations of the maximum acyclic subgraph problem do “well” on the class of graphs used to demonstrate an integrality gap of 2 for the classical linear programming relaxations. In particular, graphs from the class $G(n,p)$ with p equal to approximately $2^{\sqrt{\log n}}/n$ (and each edge (i,j) randomly directed from i to j or j to i with equal probability) were used to demonstrate an integrality gap of 2 [7]. We show that for a graph from $G(n,p)$ (with the edges directed in either direction with equal probability) where $p = \omega(1)$, our semidefinite relaxation has integrality gap at most 1.64 with high probability. The main idea of the proof is that this semidefinite relaxation provides a “good” bound on the value of a maximum acyclic subgraph if it has no small roughly balanced bisection. With high probability a random graph with uniform edge probability contains no such small balanced bisection.

References

[1] Michel X. Goemans and David P. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, Journal of the ACM **42** (1995), 1115–1145.

[2] Charles Delorme and Svatopluk Poljak, *The performance of an eigenvalue bound in some classes of graphs*, Discrete Mathematics **111** (1993), 145–156.

[3] Svatopluk Poljak, *Polyhedral and eigenvalue approximatoins of the max-cut problem*, Sets, Graphs and Numbers, Coll. Math Soc. Janos Bolyai **60** (1992), 569–581.

[4] Svatopluk Poljak and Franz Rendl, *Computing the max-cut by eigenvalues*, Discrete Applied Mathematics **62(1–3)** (1995), 249–278.

[5] Alantha. Newman, *Algorithms for string and graph layout*, Ph.D. Thesis, M.I.T. (2004).

[6] Alantha Newman, *Cuts and orderings: on semidefinite relaxations for the linear ordering problem*, Proceedings of the 7th Workshop on Approximation Algorithms (2004), 195–206.

[7] Alantha Newman and Santosh Vempala, *Fences are futile: on relaxations for the linear ordering problem*, Proceedings of the Conference on Integer Programming and Combinatorial Optimization (2001), 333-347.

Simple Cost Sharing Schemes for Multicommodity Rent-or-Buy and Stochastic Steiner Tree

Jochen Könemann

(joint work with Lisa Fleischer, Stefano Leonardi, Guido Schäfer)

1. Introduction

Multicommodity Rent-or-Buy. In the *multi-commodity rent-or-buy problem* (MRoB) we are given an undirected graph $G = (V, E)$, terminal pairs $R = \{(s_1, t_1), \ldots, (s_k, t_k)\} \subseteq V \times V$, non-negative costs c_e for all edges $e \in E$, and a parameter $M \geq 0$. The goal is to install capacities on the edges of G such that for all $(s_i, t_i) \in R$ we can simultaneously route a given amount of flow f_i from s_i to t_i. We can either *rent* capacity on an edge e at cost $\lambda(e) \cdot c(e)$, where $\lambda(e)$ is the flow traversing edge e, or *buy* infinite capacity on edge e at cost $M \cdot c(e)$. Bought edges have no incremental, flow-dependent cost. The overall objective is to find a feasible solution of smallest total cost.

The MRoB problem generalizes a number of fundamental optimization problems. For $M = \infty$, an optimum solution for an MRoB instance can be found by connecting each pair of terminals by their shortest path.

For $M = 1$, MRoB reduces to the Steiner forest problem. The *Steiner forest problem* is to compute a minimum-cost forest that contains an s_i, t_i-path for all $1 \leq i \leq k$. It is well-known that this problem is NP-hard [9] and even Max-SNP hard [7]. The best known approximation algorithm achieves a performance guarantee of $2 - 1/k$ and is due to Agrawal, Klein and Ravi [3]. Goemans and Williamson [15] generalize these results to a larger class of network design problems.

The MRoB problem is a generalization of the *single-commodity rent-or-buy* problem (SRoB). Here, in addition to the input given in an instance of the MRoB problem, one also has a *root node* $r \in V$. The root r is part of every terminal pair, i.e., $r \in \{s_i, t_i\}$ for all $1 \leq i \leq k$. Gupta et al. [11] gave a randomized 3.55-approximation algorithm for the problem.

Kumar, Gupta and Roughgarden [16] give the first constant-factor approximation algorithm for the MRoB problem. Based on the techniques used by Gupta et al. [11] for the single-commodity rent-or-buy problem, Gupta et al. [10] present a 12-approximation algorithm for the MRoB problem. Becchetti et al. [6] recently obtained the currently best known 6.828-approximation algorithm for this problem.

The MRoB problem is a special case of the *multicommodity buy-at-bulk* (MBaB) problem. The input in this problem is as in the MRoB problem, except for an

additional sub-additive monotone function $l : \mathbb{Z}^+ \to \mathbb{R}^+$. A feasible solution consists of a vector $x \in \mathbb{Z}^+_{|E|}$ of edge-capacities that allows for f_i units of flow to be routed between s_i and t_i, for all $(s_i, t_i) \in R$ simultaneously, and feasibly. The cost of installing capacities x is $\sum_{e \in E} l(x_e)c_e$ and the goal is to find a feasible capacity installation x of minimum total cost.
In [2], Awerbuch and Azar present an $O(\alpha)$-approximation for MBaB, assuming that any metric can be probabilistically approximated by a family of tree metrics with an expected distortion at most α. In [4], Bartal shows $\alpha = O(\log^2 n)$ and improves this bound in [5] to $\alpha = O(\log n \log\log n)$. More recently, Fakcharoenphol et al. [8] show that $\alpha = O(\log n)$. Recently, Andrews [1] shows that the results in [2] and [8] are best possible up to constant factors unless $\texttt{NP} \subseteq \texttt{ZPTIME}(n^{\text{polylog}(n)})$.
Stochastic Steiner Tree. The *stochastic Steiner tree problem (SST)* we consider here is the Steiner tree problem in the model of two-stage stochastic optimization with recourse. In stage one, there is a known probability distribution π on subsets of vertices and we can chose to buy a subset of edges at a given cost. In stage two, a subset of vertices T from the prior known distribution is realized, and additional edges can be bought at a possibly higher cost. The objective is to buy a set of edges in stages one and two so that all vertices in T are connected, and the expected cost is minimized.
We make no assumptions about the distribution π on the subset of vertices, except that we have access to it via a *sampling oracle*: on request, the oracle outputs a subset of vertices T drawn from the distribution. Gupta and Pál give a 12.6-approximation for SST [12]. Prior to their work, there were constant factor guarantees for the problem when all of the possible subsets realized in stage two contain a fixed root terminal [13, 14].
Common Framework. Our work uses a common framework developed in Gupta et al. [10] for MRoB and extended by Gupta and Pál in [12] for SST. This framework first chooses a random subset $S \subseteq R$ of the set of terminal pairs. It then computes an approximate Steiner forest F_S for S using an adaptation of the primal-dual algorithm for Steiner forests and buys its edges. Finally, this forest is augmented to a feasible solution for R by renting additional edges in a cheapest possible way such that all remaining terminals in $R \setminus S$ are connected.
The performance of the above framework depends strongly on a certain *stability* property of the Steiner forest algorithm used to compute F_S. For a forest F in G, let $G|F$ denote the graph resulting from contracting all trees of F. We use $c_{G|F}(u, v)$ to denote the minimum cost of any u, v-path in $G|F$. For a parameter $\beta > 0$, Gupta et al. [10] define the notion of *β-strict* algorithms for the minimum-cost Steiner forest problem:

Definition 1. *An algorithm* ALG *for the Steiner forest problem is β-*strict *if there exist cost shares ξ_{st} for all $(s, t) \in R$ such that*

(1) $\sum_{(s,t) \in R} \xi_{st} \le \mathsf{opt}_R$*, where* opt_R *is the minimum cost of a Steiner forest for R, and*
(2) $c_{G|F_{-st}}(s, t) \le \beta \cdot \xi_{st}$ *for all $(s, t) \in R$, where F_{-st} is a Steiner forest for terminal set $R_{-st} = R \setminus \{(s, t)\}$ returned by* ALG.

Gupta et al. [10] then show that using an α-approximate and β-strict Steiner forest algorithm in their framework yields an $(\alpha + \beta)$-approximation algorithm for the MRoB problem. The authors devise a 6-approximate and 6-strict algorithm for Steiner forests which yields a 12-approximate algorithm for MRoB. Their analysis can be tightened to achieve an 8-approximation. Becchetti et al. [6] reduced the approximation ratio to 6.828 by devising a $(2 + \sqrt{2})$-approximate and $(2 + \sqrt{2})$-strict primal-dual Steiner forest algorithm.
The notion of strictness defined above assumes that R is a set of terminal pairs. To extend this framework to handle SST, Gupta and Pál extend the notion of strictness to a set R of terminal subsets of arbitrary size, called *groups*. For a group $g \in R$, let $c_{G|F}(g)$ denote the minimum cost of connecting all terminals of g in $G|F$.

Definition 2. *An algorithm* ALG *for the Steiner forest problem is* β-group-strict *if there exist cost shares* ξ_g *for all* $g \in R$ *such that*

(1) $\sum_{g \in R} \xi_g \le \mathsf{opt}_R$*, where* opt_R *is the minimum cost of a Steiner forest for* R*, and*
(2) $c_{G|F_{-g}}(g) \le \beta \cdot \xi_g$ *for all* $g \in R$*, where* F_{-g} *is a Steiner forest for terminal set* $R_{-g} = R \setminus \{g\}$ *returned by* ALG.

The algorithms in [6], [10], and [12] all adapt the primal-dual Steiner forest algorithm from [3]. In these papers, strictness is achieved by adding extra edges into the Steiner forest produced by the standard primal-dual algorithm. This worsens the approximation ratio but reduces the cost of augmenting a feasible forest F_{-g} into a feasible forest for R.
Our Results. We show that the primal-dual algorithms for Steiner forest [3, 15] are 3-strict and 4-group-strict with appropriate cost sharing rules. We summarize our main contribution in the following theorem:

Theorem 3. *There exists a primal-dual* 2*-approximate algorithm for the Steiner forest problem that is* 3*-strict and* 4*-group-strict.*

This implies a 5-approximation for MRoB and a 6-approximation for SST using the framework in [10, 12]. Moreover, this also implies a 5-approximation algorithm for the 2-stage stochastic Steiner forest problem in the independent decisions model [13].
This is the first algorithm to show that the *unmodified* primal-dual Steiner forest algorithm has *constant* strict or group-strict cost shares. Finally, we present an example instance that shows that the natural primal-dual Steiner forest algorithm is not $(\frac{8}{3} - \epsilon)$-strict for any $\epsilon > 0$, therefore showing that the two-stage analysis of Gupta et al. given in [10] is nearly tight for MRoB.

References

[1] M. Andrews. Hardness of buy-at-bulk network design. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 115–124, 2004.
[2] B. Awerbuch and Y. Azar. Buy-at-bulk network design. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 542–547, 1997.

[3] A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem in networks. *SIAM J. Comput.*, 24:440–456, 1995.
[4] Y. Bartal. Probabilistic Approximation of Metric Spaces and its Algorithmic Applications. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 184–193, 1996.
[5] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proceedings, ACM Symposium on Theory of Computing*, pages 161–168, 1998.
[6] L. Becchetti, J. Könemann, S. Leonardi, and M. Pál. Sharing the cost more efficiently: Improved approximation for multicommodity rent-or-buy. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 375–384, 2005.
[7] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Inform. Process. Lett.*, 32(4):171–176, 1989.
[8] J. Fakcharoenphol, S. Rao and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings, ACM Symposium on Theory of Computing*, pages 448–455, 2003.
[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* Freeman, San Francisco, 1979.
[10] A. Gupta, A. Kumar, M. Pál, and T. Roughgarden. Approximation via cost-sharing: A simple approximation algorithm for the multicommodity rent-or-buy problem. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 606–617, 2003.
[11] A. Gupta, A. Kumar, and T. Roughgarden. Simpler and better approximation algorithms for network design. In *Proceedings, ACM Symposium on Theory of Computing*, pages 365–372, 2003.
[12] A. Gupta and M. Pál. Stochastic Steiner trees without a root. In *Proceedings, International Colloquium on Automata, Languages and Programming*, pages 1051–1063, 2005.
[13] A. Gupta, M. Pál, R. Ravi, and A. Sinha. Boosted sampling: Approximation algorithms for stochastic optimization. In *Proceedings, ACM Symposium on Theory of Computing*, pages 417–426, 2004.
[14] A. Gupta, R. Ravi, and A. Sinha. An edge in time saves nine: LP rounding approximation algorithms. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 218–227, 2004.
[15] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24:296–317, 1995.
[16] A. Kumar, A. Gupta, and T. Roughgarden. A constant factor approximation algorithm for the multicommodity rent-or-buy problem. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 333–344, 2002.

Random triangulations of planar point sets

Emo Welzl
(joint work with Micha Sharir)

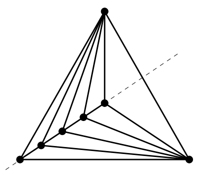
Given a set S of n points in the plane, a triangulation is a maximal crossing-free geometric graph on S (in a geometric graph the edges are realized by straight line segments). Here we consider random triangulations, where "random" refers to uniformly at random from the set of all triangulations of S. We are primarily interested in the degree sequences of such random triangulations.
To be precise, we assume that S is a set of $n + 3$ points in general position in the plane so that the convex hull of S is a triangle. For such a set and $i \in \mathbb{N}$, we let $\hat{v}_i$ denote the expected number of interior points of degree i in a random

triangulation. While—for n large enough—the number of vertices of degree 3 in a triangulation may be any integer between 0 and roughly $\frac{2n}{3}$, we show that

$$\tfrac{n}{43} \leq \hat{v}_3 \leq \tfrac{2n+3}{5}$$

and, for all $i \geq 3$, $\delta_i \in \mathbb{R}^+$ exists so that $\hat{v}_i \geq \delta_i n$, provided n is large enough. Our proofs use charging schemes among vertices in triangulations that heavily build on the structure imposed by edge flips on the set of all triangulations (see below).

General position is essential for the lower bound on $\hat{v}_3$. Consider the case where the n interior points lie on a common line containing one of the extreme points in S. Then there is a unique triangulation and this triangulation has one interior point of degree 3; hence, $\hat{v}_3 = 1$.



We relate these results to the question about the maximum and minimum possible number of triangulations in a set of n points in the plane. We show that the number of triangulations of any such set is at most 43^n, thereby improving on a previous bound of 59^n by Santos and Seidel [13]. We can also use the upper bound on $\hat{v}_3$ to infer a lower bound of roughly 2.5^n on the number of triangulations every set of $n+3$ points in general position with triangular convex hull has. However, this is inferior to the recent $\Omega(2.63^n)$-bound in [7].

Little seems to be known about random triangulations of (fixed) point sets, although the generation of random triangulations has raised some interest (see, e.g., [1, Section 4.3]). Moreover, it is a folklore open problem to determine the mixing rate of the Markov process that starts at some triangulation and keeps flipping a random flippable edge; see [10, 9] where this is treated for points in convex position. For abstract graphs there are results about random planar graphs, see, e.g., [6, 8, 5] (here one has to discriminate between the labeled and the unlabeled case).

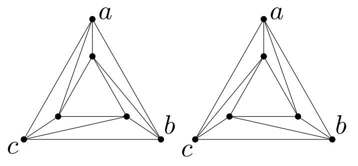
Number of Triangulations—History. David Avis was perhaps one of the first to ask whether the maximum number of triangulations of n points in the plane is bounded by c^n for some $c > 0$, see [3, page 9]. This fact was established in 1982 by Ajtai et al. [3], who show that there are at most 10^{13n} crossing-free graphs on n points—in particular, this bound holds for triangulations.

Further developments have yielded progressively better upper bounds for the number of triangulations in [16, 4, 14]. Interest in this question was also motivated by the related practical question (from geometric modeling [16]) of how many bits it takes to encode a triangulation of a point set. These investigations have so far culminated in the previously mentioned 59^n bound [13] in 2003. This compares to $\Omega(8.48^n)$, the largest known number of triangulations for a set of n points, recently derived by Aichholzer et al. [2].

For n points in convex position, the number of triangulations is known to be C_{n-2}, where $C_m := \frac{1}{m+1}\binom{2m}{m} = \Theta(m^{-3/2}4^m)$, $m \in \mathbb{N}_0$, is the mth *Catalan number.*

Other Crossing-free Graphs. Besides the intrinsic interest in obtaining bounds on the number of triangulations, they are useful for bounding the number of other kinds of crossing-free geometric graphs on a given point set, exploiting the fact that any such graph is a subgraph of some triangulation. For example, the best known upper bound on the number of crossing-free straight-edge spanning trees on a set of n points in the plane is $O((5.\dot{3}\,\tau)^n)$, if τ^n is a bound on the number of triangulations; with $\tau = 43$ this is now $O(229.\dot{3}^n)$. This follows from a result by Ribó and Rote, [11, 12], who show that any planar graph on n vertices contains at most $5.\dot{3}^n$ spanning trees. Similar results have been observed for crossing-free spanning cycles, where a bound of $O((\sqrt{6}\tau)^n) = O((2.45\,\tau)^n)$ can be obtained, as communicated by Raimund Seidel; the resulting bound of $O(105.33^n)$ falls still short of the bound of $O(86.81^n)$ for cycles given in [15], though. The total number of crossing-free planar graphs on n points is at most $2^{3n-6}\tau^n < (8\,\tau)^n$. So this is now improved to 344^n (from 472^n).

Tutte's Number of Rooted Triangulations. Let us briefly discuss a classical result from 1962 by Tutte in his *census*-series in the *Canadian Journal of Mathematics* [17]. He considers so-called *rooted triangulations*, i.e., maximal planar graphs, with a fixed face with vertices a, b, and c and n additional vertices. Two such triangulations are considered to be equal if there is an isomorphism between them, which maps each of the points a, b, and c to itself, though. The number of such triangulations is easily seen to be 1 for $n = 1$ and 3 for $n = 2$. Based on an ingenious analysis employing generating functions, Tutte shows that for $n \geq 2$ the number of such triangulations is exactly $\frac{2}{n(n+1)}\binom{4n+1}{n-1} = \Theta\left(\frac{1}{n^{5/2}}\ 9.\overline{481}^n\right)$.



How does this relate to the number of triangulations of given $n + 3$ points? On the one hand, Tutte's model counts more triangulations, because there are fewer constraints: "The interior points can be moved arbitrarily." On the other hand, distinct triangulations in the geometric setting may be equal in Tutte's; see Figure.

(Dis-)Charging. The notion of "charging" rings a bell in the context of planar graphs. The proof of the celebrated Four-Color-Theorem employs Heesch's idea of discharging (*Entladung*) in order to prove that certain configurations are unavoidable in a maximal planar graph. There one initially puts charge $6 - i$ on each vertex of degree i in a maximal planar graph—thus the overall charge is 12. Now vertices of positive charge push their charge to other vertices (they discharge) without changing the overall charge. Given that a certain set of configurations L does not occur, one proves that all vertices can discharge with a nonpositive charge in the end—a contradiction and thus the configurations in L are unavoidable.

Our scheme differs in two respects. First of all we need a quantitative version. We let every vertex have a value of $7 - i$, in this way we can make sure that the overall value in a maximal planar graph is at least n and there is at least 1 for every vertex on the average. Secondly, the "discharging" goes across a family of planar graphs, the set of all triangulations of a given point set. We show that the charge can be redistributed so that no vertex of degree exceeding 3 has positive charge, and degree-3 vertices have charge at most 43. This allows us to conclude that at least $\frac{1}{43}$ of all vertices over all triangulations have degree 3.

References

[1] O. Aichholzer, The path of a triangulation, *Proc. 15th Ann. ACM Symp. on Computational Geometry* (1999), 14–23.

[2] O. Aichholzer, T. Hackl, H. Krasser, C. Huemer, F. Hurtado, and B. Vogtenhuber, On the number of plane graphs, *Proc. 17th Ann. ACM-SIAM Symp. on Discrete Algorithms* (2006), 504–513.

[3] M. Ajtai, V. Chvátal, M. M. Newborn, and E. Szemerédi, Crossing-free subgraphs, *Annals Discrete Math.* **12** (1982), 9–12.

[4] M.O. Denny and C.A. Sohler, Encoding a triangulation as a permutation of its point set, *Proc. 9th Canadian Conf. on Computational Geometry* (1997), 39–43.

[5] O. Giménez and M. Noy, The number of planar graphs and properties of random planar graphs, Proc. International Conf. on Analysis of Algorithms, *Discrete Mathematics and Theoretical Computer Science* proc. **AD** (2005), 147–156.

[6] V.A. Liskovets, A pattern of asymptotic vertex valency distributions in planar maps, *Journal of Combinatorial Theory*, Ser. B **75** (1999), 116–133.

[7] P. McCabe and R. Seidel, New lower bounds for the number of straight-edge triangulations of a planar point set, *Proc. 20th European Workshop Comput. Geom.* (2004).

[8] C. McDiarmid, A. Steger, and D.J.A. Welsh, Random planar graphs, *Journal of Combinatorial Theory*, Ser. B **93** (2005), 187–205.

[9] L. McShine and P. Tetali, On the mixing time of the triangulation walk and other Catalan structures, *Randomization Methods in Algorithm Design*, in: *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **43** (1998), 147–160.

[10] M. Molloy, B. Reed, and W. Steiger. On the mixing rate of the triangulation walk, *Randomization Methods in Algorithm Design*, in: *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **43** (1998),179–190.

[11] A. Ribó, *Realizations and Counting Problems for Planar Structures: Trees and Linkages, Polytopes and Polyominos*, Ph.D. thesis, Freie Universiät Berlin, 2005.

[12] G. Rote, The number of spanning trees in a planar graph, *Oberwolfach Reports* **2** (2005), 969-973.

[13] F. Santos and R. Seidel, A better upper bound on the number of triangulations of a planar point set, *Journal of Combinatorial Theory*, Ser. A **102**:1 (2003), 186–193.

[14] R. Seidel, On the number of triangulations of planar point sets, *Combinatorica* **18**:2 (1998), 297–299.

[15] M. Sharir and E. Welzl, On the number of crossing-free matchings (cycles, and partitions), *Proc. 17th Ann. ACM-SIAM Symp. on Discrete Algorithms* (2006), 860–869.

[16] W.S. Smith, *Studies in Computational Geometry Motivated by Mesh Generation*, Ph. D. Thesis, Princeton University, 1989.

[17] W.T. Tutte, A census of planar triangulations, *Canadian J. of Math.* **14** (1962), 21–38.

Dynamic Routing in Graphs with Applications to Harbour Logistics

Rolf H. Möhring

(joint work with Ewgenij Gawrilow, Ekkehard Köhler, Ines Spenke, Björn Stenzel)

In modern logistic systems Automated Guided Vehicles (AGVs) are used for transportation tasks. An appropriate control of these AGVs is crucial for efficient transportation. They need to be assigned collision free routes in such a way that the throughput of goods is maximized. The determination of these routes is an online routing problem (nothing known about future requests) and also a real-time

problem, because fast answers are required (should be less than one second in practice).
We present an algorithm for this problem which avoids collisions, deadlocks, live-locks and other conflicts already at the time of route computation (conflict-free routes). We thus extend approaches of Huang, Palekar and Kapoor [2] and Kim and Tanchoco [3], respectively. In particular, we take physical properties of the AGVs into consideration in a more exact and flexible way.
The time dependent behavior of the AGVs is modeled by time-windows on the arcs of the routing graph(implicit time-expansion). Each time-window represents a free time slot at the corresponding arc depending on the routes of the AGVs that are already computed (see Fig. 1). The real-time computation for each routing request consists of the determination of a shortest path with time-windows (routing) and a subsequent readjustment of the time-windows (blocking).

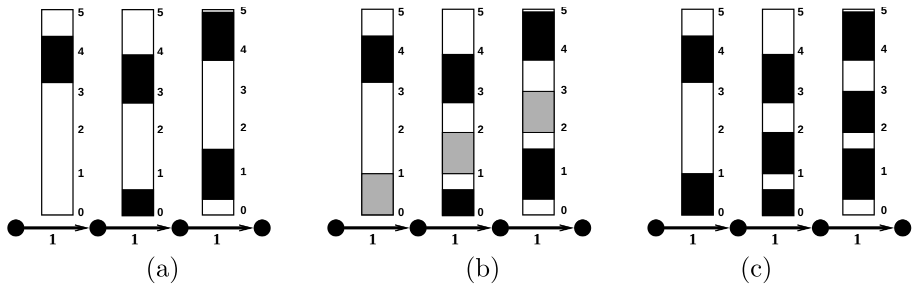


FIGURE 1. Real-time computation. (a) shows the situation before the new request arrives. There is a graph with some blockings (black) and some time-windows (white). The task is to compute a quickest path that respects the time-windows. This is illustrated in (b). The chosen path is blocked afterwards (see (c)).

The Shortest Path Problem With Time-Windows is NP-hard in general [1], but in this special case where cost correlates with elapsed time (travel time plus waiting time), our algorithm solves the problem in polynomial time (in the size of the graph and the number of time-windows) using a generalized Dijkstra algorithm algorithm that maintains an interval in each label (the expansion of such a label is shown in Fig. 2). In contrast, we can show that the problem without waiting is weakly NP-hard, while the multicommodity case turns out to be strongly NP-hard. Our routing algorithm shows very good computational times in practice and can also handle additional features such as a prescribed orientation of the AGVs at their destination. On a network with more than 30.000 arcs and up to 100 AGVs routing and blocking together take not more than some hundredth of a second on the average[1] (see Table 2).

[1]Hardware: AMD-Athlon 2100+ (1,7 Mhz) with 512 MB RAM.

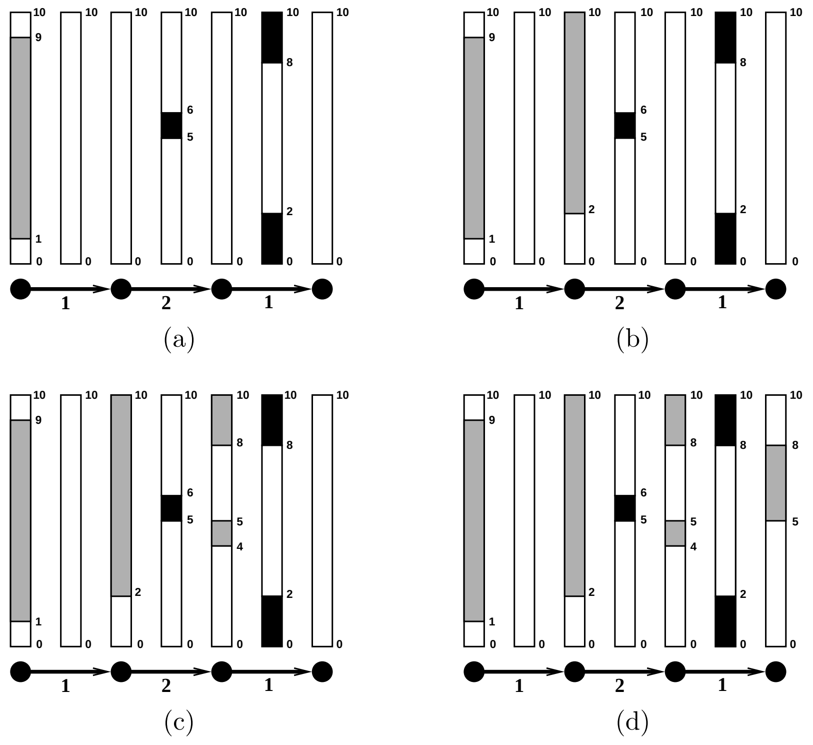


FIGURE 2. Label Expansion in the generalized Dijkstra algorithm. The label intervals are represented by grey bars (nodes). The blockings are colored black (arcs). The white intervals between these blockings are the time-windows. The figures (a) to (d) show the successive expansion of the label intervals.

In comparison with a static routing approach, in which collision avoidance is done at run time and not at route computation time, our algorithm shows a clear advantage w.r.t. total travel time for high traffic densities.

REFERENCES

[1] Desrosiers et al., (1986) *Methods for routing with time windows*, European Journal of Operational Research **23** (1986), 236–245

[2] Huang, J., Palekar, U. S., Kapoor, S., *A labeling algorithm for the navigation of automated guided vehicles*, Journal of engineering for industry **115** (1993), 315–321

[3] Kim, Ch. W., Tanchoco, J. M. A., *Conflict-free shortest-time bidirectional AGV routing*, International Journal of Production Research **29(12)** (1991), 2377–2391

Table 2. Computational times (in seconds).

Scenarios	Comp. per request		Search		Readjustment	
	maximal	⌀	maximal	⌀	maximal	⌀
25-1G-L (25 AGVs)	0.35	0.10	0.32	0.08	0.04	0.02
25-1G-S (25 AGVs)	0.14	0.06	0.11	0.04	0.03	0.02
25-2G-L (25 AGVs)	0.24	0.06	0.24	0.05	0.03	0.01
25-2G-S (25 AGVs)	0.25	0.06	0.24	0.05	0.02	0.01
25-3G-L (25 AGVs)	0.29	0.06	0.27	0.05	0.04	0.01
25-3G-S (25 AGVs)	0.23	0.06	0.18	0.05	0.04	0.01
25-4G-L (25 AGVs)	0.18	0.04	0.16	0.03	0.03	0.01
25-4G-S (25 AGVs)	0.18	0.05	0.16	0.04	0.02	0.01
50-1G-L (50 AGVs)	0.35	0.10	0.31	0.08	0.04	0.02
50-1G-S (50 AGVs)	0.23	0.07	0.20	0.05	0.04	0.02
50-2G-L (50 AGVs)	0.32	0.06	0.30	0.05	0.04	0.01
50-2G-S (50 AGVs)	0.16	0.06	0.13	0.04	0.04	0.01
100G-L (100 AGVs)	0.26	0.06	0.23	0.05	0.05	0.01
100G-S (100 AGVs)	0.23	0.06	0.20	0.04	0.05	0.01

Spanners, Weak Spanners, and Power Spanners

Christian Schindelhauer
(joint work with Klaus Volbert, Martin Ziegler)

For $c \in \mathbb{R}$, a *c-spanner* is a subgraph of a complete Euclidean graph satisfying that between any two vertices there exists a path of weighted length at most c times their geometric distance. Based on this property to approximate a complete weighted graph, sparse spanners have found many applications, e.g., in FPTAS, geometric searching, and radio networks. In a *weak* c-spanner, this path may be arbitrary long but must remain within a disk of radius c-times the Euclidean distance between the vertices. Finally in a *c-power* spanner, the energy consumed on such a path must be at most c-times the one consumed on a direct link. Such graphs have important implications for wireless networks regarding congestion and energy efficiency.
While it is known that any c-spanner is also both a weak C_1-spanner and a C_2-power spanner (for appropriate C_1, C_2 depending only on c but not on the graph under consideration), we show that the converse fails: There exists a family of c_1-power spanners that are no weak C-spanners and also a family of weak c_2-spanners that are no C-spanners for any fixed C (and thus no uniform spanners, either). However the deepest result of the present work reveals that any weak spanner *is* also a uniform power spanner. We further generalize the latter notion by considering (c, δ)-power spanners where the sum of the δ-th powers of the

lengths has to be bounded; so $(\cdot, 2)$-power spanners coincide with the usual power spanners and $(\cdot, 1)$-power spanners are classical spanners. Interestingly, these $(\cdot, \delta)$-power spanners form a strict hierarchy where the above results still hold for any $\delta \geq 2$; some even hold for $\delta > 1$ while counterexamples exist for $\delta < 2$. We show that every self-similar curve of fractal dimension $d > \delta$ is no (C, δ)-power spanner for any fixed C, in general.

References

[1] Christian Schindelhauer, Klaus Volbert, Martin Ziegler, *Spanners, Weak Spanners, and Power Spanners*, Proceedings of the 15th Annual International Symposium on Algorithms and Computation (ISAAC 04), 805-821, 2004

Rainbow colorings of hypercubes

Arnfried Kemnitz
(joint work with Maria Axenovich, Heiko Harborth, Meinhard Möller, and Ingo Schiermeyer)

Let Q_n be a hypercube of dimension n, that is, a graph whose vertices are binary n-tuples and two vertices are adjacent if and only if the corresponding n-tuples differ in exactly one position. A coloring of the edges of a graph H is called *rainbow* if no two edges of H have the same color, that is, in a rainbow coloring the edges are totally multicolored. Given a host graph G and a subgraph $H \subseteq G$, a coloring of the edges is called *H-anti-Ramsey* iff every copy of H in G has at least two edges of the same color. Denote by $f(G, H)$ the maximum number of colors such that there is no rainbow copy of H in some coloring of the edges of G with $f(G, H)$ colors (which is of course the maximum number of colors to be used in an H-anti-Ramsey coloring of G). Equivalently, any coloring of the edges of G with at least $rb(G, H) = f(G, H) + 1$ colors contains a rainbow copy of H. The number $rb(G, H)$ will be called *rainbow number* and $f(G, H)$ *anti-Ramsey number* of graphs G and H.

This anti-Ramsey problem was introduced by Erdős, Simonovits and Sós [1]. They conjectured $rb(K_n, C_p) = n\big(\frac{p-2}{2} + \frac{1}{p-1}\big) + O(1)$ which was recently proved by Montellano-Ballesteros and Neumann-Lara [2].

One of the less studied directions in this area is the investigation of the anti-Ramsey function $f(G, H)$ when G is not complete. We present some results for $f(G, H)$ in case that G and H are hypercubes Q_n and Q_k.

The following result provide general lower and upper bounds for $f(Q_n, Q_k)$.

Theorem 1.

$$n\,2^{n-1} - \Big\lfloor \frac{n}{k}\big(2^{n-1} - k + 1\big) \Big\rfloor \leq f(Q_n, Q_k) \leq n\,2^{n-1}\bigg(1 - \frac{n-k}{(n-1)k2^{k-2}}\bigg).$$

For fixed k, the lower and upper bounds on $f(Q_n, Q_k)/(n2^{n-1})$ by Theorem 1 tend to $1 - \frac{1}{k}$ and $1 - \frac{1}{k2^{k-2}}$, respectively, as $n \to \infty$.
Setting $k = 2$ in Theorem 1 we immediately obtain the following result.

Corollary 2.

$$n\,2^{n-2} + \left\lceil \frac{n}{2} \right\rceil \le f(Q_n, Q_2) \le (n+1)\,2^{n-2}.$$

For $k = n - 1$ we determine the exact value of $f(Q_n, Q_k)$. It turns out that all edges except three can be colored differently if $n \ge 6$.

Theorem 3. $f(Q_n, Q_{n-1}) = \begin{cases} n\,2^{n-1} - 4 & \textit{for} \quad n = 3, 4, 5, \\ n\,2^{n-1} - 3 & \textit{for} \quad n \ge 6. \end{cases}$

¿From Corollary 2 we obtain $18 \le f(Q_4, Q_2) \le 20$. The lower bound is attained.

Theorem 4. $f(Q_4, Q_2) = 18$.

With a lot of case analysis we could determine the exact value of $f(Q_5, Q_3)$.

Theorem 5. $f(Q_5, Q_3) = 68$.

References

[1] P. Erdős, M. Simonovits and V.T. Sós, *Anti-Ramsey theorems*, Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday), Vol. II, pp. 633–643. Colloq. Math. Soc. Janos Bolyai, Vol. 10, North-Holland, Amsterdam, 1975.

[2] J.J. Montellano-Ballesteros and V. Neumann-Lara, *An Anti-Ramsey Theorem on Cycles,* Graphs and Combinatorics **21** (2005), 343–354.

Graphs, Games and Algorithms

Paul G. Spirakis

The theory of noncooperative strategic games has had a remarkable growth in the last 50 years. However, its intersection with Computer Science remained remarkably small till 5-6 years ago. This has changed recently due to the efforts of (mainly) CS theorists to understand the effect of antagonism and cooperation on the growth and dynamics of the Internet. As a result, a new field has been born, i.e. that of Algorithmic Game Theory. A natural consequence of this fact is the appearance of an interaction between Game Theory and Algorithmic Graph Theory.
We have encountered examples of such an interaction in our examination of network congestion games, where the pure strategies of the players are paths in a directed graph. Some of our results indicate that the structure of the network graph is crucial for (a) the efficient decision of existence of e.g. Pure Nash Equilibria and (b) their efficient computation when they exist. In fact, we have already a partial characterization of network graphs for which a sequence of best responses converges fast to a pure equilibrium.

The role of graph structure with respect to equilibria existence and finding seems to be strong also in games that are not in the class of congestion games. We have some evidence of this in situations of confrontation-type games in graphs.
An important class of games where the welfare of a player depends directly only on a few other players is that of Graphical Games (defined by Kearns, Littman and Singh). Here the graph models the dependencies between each player's payoff and the strategies of the other players that affect this player directly. Quite surprisingly, the existence and the finding of Pure Nash Equilibria in such games strongly relates to the clique structure of such dependency graphs.
A major part of our current research is the examination of such connections of the Complexity of pure (and also mixed) equilibria and the structure of graphs associated to the strategic game.

Tutte sets: algorithmic and structural aspects

HAJO BROERSMA
(joint work with Doug Bauer, Nathan Kahl, Aurora Morgana, Ed Schmeichel, Thomas Surowiec)

A well-known formula of Tutte and Berge expresses the size of a maximum matching in G in terms of what is usually called the deficiency of G. A subset X of G for which this deficiency is attained is called a Tutte set of G. While much is known about maximum matchings, less is known about the structure of Tutte sets. In this report, we study the structural and algorithmic aspects of maximal Tutte sets in a graph G. For details we refer to [1] and [2].

1. Introduction and preliminary results

Given a graph G, let $\omega(G)$ (resp., $\omega_0(G)$) denote the number of components (resp., odd components) of G. An important result in matching theory is due to Tutte [8].

Theorem 1. *(Tutte's Theorem) A graph G has a perfect matching if and only if $\omega_0(G - X) \leq |X|$ for all $X \subset V(G)$.*

In 1958, Berge [4] extended Tutte's Theorem to give the exact size of a maximum matching in a graph G. Define the **deficiency** of G, denoted def(G), by $\max_{X \subset V(G)}\{\omega_0(G - X) - |X|\}$, where the maximum is taken over all proper subsets of $V(G)$. It can be shown that def(G) is the number of vertices unmatched by a maximum matching in G, and thus we have the following.

Theorem 2. *(Tutte-Berge Formula) The maximum size of a matching in a graph G is $\frac{|V(G)|-def(G)}{2}$.*

Motivated by the above formula, we define a **Tutte set** (called barrier in [7]) in G as a proper subset $X \subset V(G)$ such that $\omega_0(G - X) - |X| = \text{def}(G)$.

Although much is known about matchings in graphs, less is known about Tutte sets. In this report, we will be particularly interested in the structure of maximal Tutte sets. By the well-known Edmonds-Gallai decomposition, studying maximal Tutte sets in general graphs reduces to the study of maximal Tutte sets in graphs with perfect matchings.

2. The structure of maximal Tutte sets

In this part of the report we study the structure of Tutte sets and their relation to extreme sets and independent sets in D-graphs, which will be introduced shortly. We give two characterizations of maximal Tutte sets. One characterization involves the closely related concept of an extreme set in a graph, introduced in [7].
Let $X \subset V(G)$. It is immediate that $\mathrm{def}(G-X) \leq \mathrm{def}(G)+|X|$. We call $X \subset V(G)$ an **extreme set** in G if $\mathrm{def}(G-X) = \mathrm{def}(G) + |X|$. It is easy to see that every Tutte set of G is an extreme set, but not conversely.
However we have the following result, which implies Lemma 3.3.8 in [7].

Theorem 3. *Let G be a graph. A maximal extreme set in G is a maximal Tutte set in G, and conversely.*

The definitions of the concepts of a Tutte set and of an extreme set both involve a technical equality related to the matching structure of the graph. These concepts are related as we have seen above. In order to relate them to the easier, e.g. less technical, concept of an independent set, we next introduce the auxiliary concept of the D-graph, $D(G)$, of a graph G. Assume for now that G contains a perfect matching M. The graph $D(G)$ is then defined as follows: $V(D(G)) = V(G)$, and $E(D(G)) = \{(x, y) \mid G-\{x, y\}$ contains a perfect matching $\}$. In particular, every edge in M is an edge in $D(G)$.
It will be useful to give an alternative definition of $E(D(G))$. Let M be a perfect matching in G. We denote by $P_M[x, y]$ an M-alternating-path in G joining x and y, which begins and ends with an edge in M. (Similarly, we denote by $P_M(x, y)$ an M-alternating-path in G joining x and y, which begins and ends with an edge not in M; the M-alternating-paths $P_M[x, y)$ and $P_M(x, y]$ are defined analogously). By a theorem of Berge [3], $(x, y) \in E(D(G))$ if and only if there exists a path $P_M[x, y]$ in G. One easily checks that this definition of $E(D(G))$ is independent of the choice of the perfect matching M.
We next establish an important connection between extreme sets in G and independent sets in $D(G)$.

Theorem 4. *Let G be a graph with a perfect matching, and let $X \subseteq V(G)$. Then X is an extreme set in G if and only if X is an independent set in $D(G)$.*

Combining Theorems 3 and 4, we obtain

Theorem 5. *Let G contain a perfect matching, and let $X \subseteq V(G)$. Then the following are equivalent.*

- *X is a maximal Tutte set in G*
- *X is a maximal extreme set in G*

- *X is a maximal independent set in $D(G)$.*

We note that we can extend Theorem 5 to graphs which do not contain a perfect matching by using the Edmonds-Gallai decomposition of G. In the remaining part of the report we concentrate on graphs that have a perfect matching.

3. Iterated D-graphs

The results of the previous section show that there is an intimate relation between Tutte sets and extreme sets in graphs and independent sets in D-graphs. Motivated by this, in this section we will proceed with studying D-graphs and iterated D-graphs. Let us take $D^0(G) = G$, and recursively define $D^k(G) = D(D^{k-1}(G))$, for $k \geq 1$. We begin with an easy observation.

Theorem 6. *For any graph G with a perfect matching, G is isomorphic to a spanning subgraph of $D(G)$, denoted $G \preceq D(G)$.*

Corollary 7. *For any graph G with a perfect matching, $G \subseteq D^2(G)$.*

Consequently, $D^k(G) \preceq D^{k+1}(G)$, for any $k \geq 0$. This, together with the fact that each $D^k(G)$ has vertex set $V(G)$, implies $D^l(G) \cong D^{l+1}(G)$, for some finite integer l. We define the **level** of G (denoted level(G)) to be the smallest such integer l.
In order to characterize graphs G with $D^{k+1}(G) \cong D^k(G)$, $k \in \{0, 1, 2\}$, we introduce the following concepts.
Let M be a perfect matching in G. If for all edges $[a, a^{'}] \in M$ and all (not necessarily distinct) paths $P_M(x, a), P_M(a^{'}, y)$ in G, with $x, a, a^{'}, y$ distinct, we have either $[x, y] \in M$ or the edge (x, y) in G, we call G **edge-closing**; if we always have either $[x, y] \in M$ or a path $P_M(x, y)$ in G, we call G **path-closing**; if we always have either $[x, y] \in M$ or a walk $W_M(x, y)$ in G, we call G **walk-closing**. (Since $P_M(x, a)$ o $[a, a^{'}]$ o $P_M(a^{'}, y)$ is already a walk $W_M(x, y)$, we see that every graph is trivially walk-closing.)
Our main result in this section is that

(1) $D(G) \cong G$ if and only if G is edge-closing
(2) $D^2(G) \cong D(G)$ if and only if G is path-closing
(3) $D^3(G) \cong D^2(G)$ if and only if G is walk-closing.

The last statement implies that for every graph G with a perfect matching, $D^3(G) \cong D^2(G)$.

4. Characterizing D-graphs

Recall that a graph G is a D-graph if $G = D(H)$, for some graph H. It would be interesting to find a good characterization of D-graphs. In fact it would suffice to characterize level 1 D-graphs, since a level 0 graph is always a D-graph (of itself), and a level 2 graph is never a D-graph (or H would have level 3).
Although finding a good characterization of D-graphs may be difficult in general, it is easy for the class of bipartite graphs.

Theorem 8. *Let G be a bipartite graph with a perfect matching. Then G is a D-graph if and only if level(G) $= 0$.*

In particular, a tree T is a D-graph if and only if level(T) $= 0$ (i.e., $D(T) \preceq T$, hence $D(T) \cong T$). It is easy to characterize trees T with $D(T) \cong T$.

Theorem 9. *Let T be a tree on n vertices with a perfect matching. Then $D(T) \cong T$ if and only if T has the form of a tree $T^{'}$ on $n/2$ vertices and a $\overline{K_{n/2}}$ joined by a perfect matching.*

5. Algorithmic results on Tutte sets

In this part of the report we consider the problem of finding maximal or maximum Tutte sets in graphs. Whereas the first problem is polynomially solvable, the second problem is NP-hard in general but polynomially solvable for graphs with level 0 or level 1.
Consider the following decision problem.

MAXIMUM TUTTE SET
INSTANCE: Graph G, integer $k \geq 0$.
QUESTION: Does G contain a Tutte set X with $|X| \geq k$?

Theorem 10. *MAXIMUM TUTTE SET is NP-complete.*

We used a polynomial reduction from the following well-known NP-complete problem [6].

INDEPENDENT SET
INSTANCE: Graph G, integer $k \geq 0$.
QUESTION: Is $\alpha(G) \geq k$?

We first thought that a larger connectivity would cause the problem to become polynomially solvable, but this is not the case.

Theorem 11. *MAX TUTTE SET is NP-complete for k-connected graphs, for any $k \geq 1$.*

Since INDEPENDENT SET is NP-complete for the class of 2-connected planar graphs with a perfect matching, we also get:

Theorem 12. *MAX TUTTE SET is NP-complete for the class of 2-connected planar graphs.*

Since INDEPENDENT SET remains NP-complete for triangle-free graphs [6], we also get:

Theorem 13. *MAX TUTTE SET is NP-complete for triangle-free graphs.*

In contrast to the NP-completeness results, we now consider several interesting classes of graphs in which maximum Tutte sets can be found in polynomial time. We first note the following result.

Theorem 14. *MAX TUTTE SET $\in P$ for the class of graphs with level 0 or 1.*

A graph G is called **elementary** if it contains a perfect matching and if the edges which occur in a perfect matching in G induce a connected subgraph.
The following can be deduced from results in [7].

Theorem 15. *A graph G is elementary if and only if $D(G)$ is a complete multipartite graph.*

Since finding a maximum independent set in a complete multipartite graph is trivial, we can obtain the following result.

Theorem 16. *MAX TUTTE SET $\in P$ for the class of elementary graphs.*

A graph G is called **1-tough** if $\omega(G - X) \leq |X|$ for all non-empty $X \subseteq V(G)$. We obtained the following result.

Theorem 17. *MAX TUTTE SET $\in P$ for the class of 1-tough graphs.*

Corollary 18. *MAX TUTTE SET $\in P$ for the following classes of graphs:*

(a) *hamiltonian graphs,*
(b) *2-connected claw-free graphs,*
(c) *k-regular, k-edge-connected graphs, for any $k \geq 1$.*

6. Open problems

We conclude with some open problems.

(1) We know that MAX TUTTE SET can be solved in polynomial time for graphs of level 0 or 1, elementary graphs, and 1-tough graphs. Are there other interesting classes of graphs for which MAX TUTTE SET can be solved in polynomial time?
(2) We know that MAX TUTTE SET is NP-complete for 2-connected planar graphs and polynomial for 4-connected planar graphs. What is the complexity of MAX TUTTE SET for 3-connected planar graphs?
(3) MAX TUTTE SET can be solved in polynomial time for 1-tough graphs, and hence for planar 1-tough graphs. Given $\epsilon > 0$, is MAX TUTTE SET polynomial for planar $(1 - \epsilon)$-tough graphs? We think we can prove that MAX TUTTE SET is NP-complete for $(1 - \epsilon)$-tough general graphs.
(4) The class of D-graphs has been useful in our study of Tutte sets. But it remains an open problem whether level 1 D-graphs can be recognized in polynomial time (the problem is uninteresting for level 0 or 2 graphs). This recognition problem becomes trivial for the class of bipartite graphs.

References

[1] D. Bauer, H. J. Broersma, N. Kahl, A. Morgana, E. Schmeichel, and T. Surowiec. Tutte sets in graphs II: the complexity of finding maximum Tutte sets. *Preprint*, 2005.
[2] D. Bauer, H. J. Broersma, A. Morgana, and E. Schmeichel. Tutte sets in graphs I: maximal Tutte sets and D-graphs. *Preprint*, 2005.
[3] C. Berge. Two theorems in graph theory. *Proc. Nat. Acad. Sci. USA*, 43:842 – 844, 1957.
[4] C. Berge. Sur le couplage maximum d'un graphe. *C. R. Acad. Sci. (Paris)*, 247:258 – 259, 1958.
[5] J. R. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *J Res Nat Bur Standards*, pages 125 – 130, 1965.
[6] M. R. Garey and D. S. Johnson. *Computers and Intractability.* Freeman, San Francisco, CA, 1979.
[7] L. Lovász and M. D. Plummer. Matching theory. In *Ann. Discrete Math.* North-Holland, Amsterdam, 1986.
[8] W. T. Tutte. The factorization of linear graphs. *J. London Math. Soc.*, 22:107 – 111, 1947.

On counting homomorphisms to directed acyclic graphs

Martin Dyer
(joint work with Leslie Ann Goldberg, Mike Paterson)

We give a dichotomy theorem for the problem of counting homomorphisms to directed acyclic graphs. H is a fixed directed acyclic graph. The problem is, given an input digraph G, how many homomorphisms are there from G to H. We give a graph-theoretic classification, showing that for some digraphs H, the problem is in P and for the rest of the digraphs H the problem is #P-complete. An interesting feature of the dichotomy, which is absent from related dichotomy results, is that there is a rich supply of tractable graphs H with complex structure. Our result is a dichotomy theorem for the problem of counting homomorphisms to directed acyclic graphs.
A *homomorphism* from a (directed) graph $G = (V, E)$ to a (directed) graph $H = (\mathcal{V}, \mathcal{E})$ is a function from V to $\mathcal{V}$ that preserves (directed) edges. That is, the function maps every edge of G to an edge of H.
Hell and Nešetřil [7] gave a dichotomy theorem for the *decision* problem for undirected graphs H. In this case, H is an undirected graph (possibly with self-loops). The input, G, is an undirected simple graph. The question is "Is there a homomorphism from G to H. They showed that the decision problem is in P if the fixed graph H has a loop, or is bipartite. Otherwise, it is NP-complete. Dyer and Greenhill [3] established a dichotomy theorem for the corresponding *counting* problem in which the question is "How many homomorphisms are there from G to H". They showed that the problem is in P if every component of H is either a complete graph with all loops present or a complete bipartite graph with no loops present[1]. Otherwise, it is #P-complete. Bulatov and Grohe [1] extended the

[1]The graph with a singleton isolated vertex is the degenerate complete bipartite graph with no loops.

counting dichotomy theorem to the case in which H is an undirected *multigraph.* Their result will be discussed in more detail below.
In this paper, we consider the related counting problem for *directed* graphs. First, consider the corresponding decision problem. H is a fixed digraph, and the problem, given an input digraph G, is "Is there a homomorphism from G to H?" It is conjectured [8, Conjecture 5.12] that there is a dichotomy theorem for this problem, in the sense that, for every H, the problem is either polynomial-time solvable or NP-complete. (See also [2].) Currently, there is no graph-theoretic conjecture stating what the two classes of digraphs will look like. Obtaining such a dichotomy may be difficult. Indeed (see [Theorem 5.14][8]), Feder and Vardi have shown [5, 6] that the resolution of the dichotomy conjecture for digraphs would also resolve their long-standing dichotomy conjecture for constraint satisfaction problems. There are some known dichotomy classifications for restricted classes of digraphs, however, the problem is open [8] even when H is restricted to be an oriented tree.
In this paper, we give a dichotomy theorem for the corresponding counting problem, for the class of digraphs in which H can be any *acyclic* directed graph. An interesting feature of this problem, which is different from the previous dichotomy theorems that we have mentioned, is that there is a rich supply of tractable graphs H with complex structure.
The formal statement of our dichotomy is given below. Here is an informal description. First, the problem is #P-complete unless H is a *layered* digraph, meaning that the vertices of H can be arranged in levels, with edges going from one level to the next. We show that the problem is in P for a layered digraph H if the following condition is true (otherwise it is #P-complete). The condition is that, for every pair of vertices x and x' on level i and every pair of vertices y and y' on level $j > i$, the product of the graphs $H_{x,y}$ and $H_{x',y'}$ is isomorphic to the product of the graphs $H_{x,y'}$ and $H_{x',y}$. The details of the product that we use (from [4]) are given below. The notion of isomorphism is the usual (graph-theoretic) one, except that certain short components are dropped, as described below. The precise definition of $H_{x,y}$ is given below, but the reader can think of it as the subgraph between vertex x and vertex y. Some fairly complex graphs H satisfy this condition, and for these graphs H the counting problem is in P.
The hardness side of our dichotomy proof relies on two fundamental results of Bulatov and Grohe [1] and Lovász [9], and on the *layered cross product* (LCP) [4] of layered digraphs. Our algorithm for counting graph homomorphisms for tractable digraphs H is based on factoring under the LCP. A difficulty is that the relevant algebra lacks unique factorisation. We deal with this by first multiplying by "preconditioners".

References

[1] A. Bulatov and M. Grohe, The complexity of partition functions, in *Automata, Languages & Programming: 31st International Colloquium*, Lecture Notes in Computer Science **3142**, pp. 294–306, 2004.

[2] A. Bulatov and V. Dalmau, Towards a dichotomy theorem for the counting constraint satisfaction problem, in *Proc. 44th IEEE Symposium on Foundations of Computer Science*, IEEE, pp. 562–572, 2003.
[3] M. Dyer and C. Greenhill, The complexity of counting graph homomorphisms, *Random Structures & Algorithms* **17** (2000), 260–289.
[4] S. Even and A. Litman, Layered cross product: a technique to construct interconnection networks. *Networks* **29** (1997), 219–223.
[5] T. Feder and M. Vardi, Monotone monadic SNP and constraint satisfaction, 25th STOC 1993 612–622.
[6] T. Feder and M. Vardi, The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory, *SIAM J. Comput.* (28) (1998) 57–104.
[7] P. Hell and J. Nešetřil, On the complexity of H-coloring, *Journal of Combinatorial Theory Series B* **48** (1990), 92–110.
[8] P. Hell and J. Nešetřil, *Graphs and homomorphisms*, Oxford University Press, 2004.
[9] L. Lovász, Operations with structures, *Acta. Math. Acad. Sci. Hung.*, **18** (1967), 321–328.

Faster approximation algorithms for covering and packing problems

Daniel Bienstock

During the last twenty-plus years we have witnessed many significant advances in the theory and practice of linear programming. These advances, reinforced by large improvements in computing platforms, have lead to major speedups in the solution of linear programs. It is routinely claimed that we can now solve large linear programs thousands of times faster than just fifteen years ago [5].

Nevertheless, difficult linear programs remain. The prototypical example of a difficult linear program arises in the context of *network routing* applications. Here, we are given a network with capacities, and a list of "point-to-point" demands to be routed. The routing allows fractional amounts of flow to be sent along edges of the network, but the key constraint is that all the demands must be routed simultaneously without exceeding the capacities.

In one class of versions of this problem, we are given a per-unit flow *cost* for each edge of the network: here we must route all demands, constrained as in the preceding paragraph, at minimum cost. This is nothing other than the standard minimum-cost multicommodity flow problem [1].

A version of the problem of perhaps greater practical impact (at least in networking applications) is the so-called *maximum concurrent flow problem.*Here we know in advance that it is impossible to route all demands without exceeding capacities. Instead, we would like to route a common fraction, θ, of every demand while not exceeding capacities – and we would like to maximize θ. The maximum, θ^*, is called the *throughput* of the network.

In experiments reported in [2] it was shown that the running time of state-of-the-art commercial implementations of linear programming algorithms, can grow *cubically* with the number of variables, when running concurrent flow problems. On networks with roughly one thousand nodes, several thousand edges, and where (say) half of the nodes are source nodes for demands, the running time becomes

prohibitively expensive (of the order of months of CPU time) even on a fast computer. An additional complicationis the large amount of memory required by commercial solvers.
These facts are not new – at least in qualitative form, they have been known in the networking community for at least thirty years. It is no surprise that the first algorithmic attempt at circumventing the difficult linear programs arose from this community. This is the so-called *flow deviation method* [6] for the maximum concurrent flow problem. This method uses a combination of two simple ideas: first, it maintains a strictly feasible flow, and, at each iteration, it increases throughput by scaling all flows by a common constant. The new flow will be more congested – we "decongest" it by solving a nonlinear programming problem. This is the second idea. Where u_{ij} is the capacity of arc (i, j), and f_{ij} is the total combined multicommodity flow on arc (i, j), the nonlinear function to be minimized is

$$\sum_{(i,j)} \frac{f_{ij}}{u_{ij} - f_{ij}}.$$

This function is minimized using a first-order (Frank-Wolfe) algorithm, starting from the current flow, and subject to keeping the throughput constant. The minimizer, or an approximate minimizer, will have substantially lower congestion levels, and the scaling step can now be repeated.
A modern analysis of this algorithm [4] shows that it has good theoretical properties: for any $\epsilon > 0$ it computes a flow with throughput at least $(1 - \epsilon)theta^*$ in a number of iterations that grows (at most) proportional to ϵ^{-2} and polynomial factors. The flow deviation method remains popular in the networking community because it is "lightweight" and simple to implement: each iteration can be reduced to a number of shortest path computations.
Shahrokhi and Matula [18] independently developed a different methodology for the concurrent flow problem. In their approach, one routes 100% of each commodity, and instead the maximum *load* of any edge is to be minimized. Here, the load of an edge (i, j) under a multicommodity flow f equals f_{ij}/u_{ij}. It can be seen that the maximum throughput problem and the minimum load problem are equivalent (maximum throughput = inverse of minimum load). Sharokhi and Matula's approach seeks to minimize a potential function of the form

$$\sum_{(i,j)} exp(\alpha \frac{f_{ij}}{u_{ij}}).$$

Their approach is closely related to first-order methods and also entails shortest path computations. They showed that their algorithm has complexity $O(\epsilon^{-5})$ times polynomial factors.
Following Sharokhi and Matula's work, a large volume of research was produced by the theoretical computer science community (see [12], [9], [10], [11],[16], [15], [17], [8], [7] among others). This work gradually improved and generalized the results of [18], in the end producing algorithms that require $O(\epsilon^{-2})$ iterations for solving minimum cost "fractional packing" or "covering" problems, a significant

generalization over multicommodity flow problems. Further, these methods can prove efficient in practice [2].
A major contribution in this area has recently been made by Nesterov [14]. To cast his result in the context of this abstract, Nesterov produced an algorithm for the concurrent flow problem that generates, for any $\epsilon > 0$, a flow with throughput at least $\theta^* - \epsilon$, by solving a sequence of separable convex quadratic minimum-cost flow problems. The number of iterations in Nesterov's algorithm grows as ϵ^{-1} – times possibly non-polynomial factors. Also, notice that *a priori* we do not know the order of magnitude of θ^*, so a *relative error* guarantee is what is needed.
By adapting some techniques from [9], recently [3] we were able to improve this result so that (a) the approximate flow has throughput at least $(1-\epsilon)\theta^*$ (i.e., a relative error guarantee) *and* the number of iterations grows proportional to ϵ^{-1} times polynomial factors, thus matching a lower bound of Khachiyan. Furthermore, each iteration can be reduced to a number of shortest path computations.
Some recent work of Nemirovski [13] appears promising. It may be the case that this work can also be adapted to produce provably good approximation algorithms.

References

[1] R. Ahuja, T. L. Magnanti, and J. Orlin, *Networks Flows: Theory, Algorithms, and Practice*, Prentice Hall. 1993.

[2] D. Bienstock, *Potential Function Methods for Approximately Solving Linear Programming Problems, Theory and Practice*, Kluwer, Boston. 2002.

[3] D. Bienstock and G. Iyengar. *Solving fractional packing problems in O*(1/epsilon) iterations. Proc. 26th Ann. Symp. Theory of Computing* (Chicago, 2004) 146-155.
Asymptotic analysis of the flow deviation method for the maximum concurrent flow problem. Math. Prog. **91**:379–492, 2002.

[4] D. Bienstock and O. Raskina. *Asymptotic analysis of the flow deviation method for the maximum concurrent flow problem. Math. Prog.* **91**:379–492, 2002.

[5] R. Bixby, personal remark.

[6] L. Fratta, M. Gerla and L. Kleinrock, *The flow deviation method: an approach to store-and-forward communication network design. Networks* **3**:97-133, 1971.

[7] L@.K. Fleischer. *Approximating fractional multicommodity flow independent of the number of commodities. SIAM J. Disc. Math.*, **13**:505-520, 2000.

[8] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *Proc. 39th Ann. Symp. on Foundations of Comp. Sci. (FOCS)*, 300-309, 1998.

[9] M.D. Grigoriadis and L.G. Khachiyan. *Fast approximation schemes for convex programs with many blocks and coupling constraints. SIAM J. Optim.*, **4**:86-107, 1994.

[10] M.D. Grigoriadis and L.G. Khachiyan. *An exponential-function reduction method for block-angular convex programs. Networks* **26**:59-68, 1995.

[11] M.D. Grigoriadis and L.G. Khachiyan. *Approximate minimum-cost multicommodity flows in $\tilde{O}(\epsilon^{-2}KNM)$ time. Math. Prog.*, **75**:477-482, 1996.

[12] T. Leighton, F. Makedon, S. Plotkin, C. Stein, E. Tardos and S. Tragoudas. *Fast approximation algorithms for multicommodity flow problems,* Proc. *23nd Ann. ACM Symp. on Theory of Computing*, 101-111, 1999.

[13] A. Nemirovski. *Prox-method with rate of convergence $O(\frac{1}{t})$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems.* (2003)

[14] Y. Nesterov. *Smooth minimization of non-smooth functions.* CORE Discussion Paper, CORE, UCL, Belgium. (2003).
[15] S. Plotkin and D. Karger. *Adding multiple cost constraints to combinatorial optimization problems, with applications to multicommodity flows.* In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, 18-25, 1995.
[16] S. Plotkin, D.B. Shmoys and E. Tardos. *Fast approximation algorithms for fractional packing and covering problems,* Math. *Oper. Res.* **20**:495-504, 1995.
[17] T. Radzik. *Fast deterministic approximation for the multicommodity flow problem. Proc. 6th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 1995.
[18] F. Shahrokhi and D.W. Matula. *The maximum concurrent flow problem. J. ACM* **37**:318-334, 1991.

Phylogenetic trees and 3- and 4-leaf powers

Andreas Brandstädt
(joint work with Van Bang Le and R. Sritharan)

Motivated by the search for underlying phylogenetic trees, Nishimura, Ragde and Thilikos [5] defined the notion of *k-leaf power* as follows: An undirected finite graph G is the *k-leaf power* of a tree T if its vertices are leaves of T such that two vertices are adjacent in G if and only if their distance in T is at most k. Then T is a *k-leaf root* of G. In [5], a (very complicated) $O(n^3)$ time recognition algorithm for 3- and for 4-leaf powers is given. For $k \geq 5$, characterization and recognition of k-leaf powers is an open problem.
Recently, Dom, Guo, Hüffner, Niedermeier [3] characterized 3-leaf powers in terms of forbidden subgraphs. We show in [1] that a connected graph is a 3-leaf power if and only if it results from a tree by substituting cliques into its vertices. Another characterization in [1] leads to linear-time recognition of 3-leaf powers.
Moreover, Rautenbach [6] (see also [4]) characterized 4-leaf powers without true twins in terms of forbidden subgraphs. We give new characterizations for 4-leaf powers and, as a byproduct and important tool, for squares of trees. As a consequence, we obtain linear time recognition of 4-leaf powers.

References

[1] A. Brandstädt, V.B. Le, *Structure and linear time recognition of* 3*-leaf powers*, accepted for Information Processing Letters.
[2] A. Brandstädt, V.B. Le, R. Sritharan, *Structure and linear time recognition of* 4*-leaf powers*, Submitted for publication.
[3] M. Dom, J. Guo, F. Hüffner, R. Niedermeier, *Error compensation in leaf root problems*, In: Proceedings of 15th ISAAC, LNCS 3341, 389-401, 2004; To appear in Algorithmica.
[4] M. Dom, J. Guo, F. Hüffner, R. Niedermeier, *Extending the tractability border for closest leaf powers*, In: Proceedings of 31st Workshop on Graph-Theoretic Concepts in Computer Science WG 2005, LNCS 3787, 397–408, 2005.
[5] N. Nishimura, P. Ragde, D.M. Thilikos, *On graph powers for leaf-labeled trees*, J. Algorithms **42** (2002), 69–108.
[6] D. Rautenbach, *Some remarks about leaf roots*, manuscript (2004).

Simple Coresets for Clustering Problems

Christian Sohler
(joint work with Gereon Frahling)

Clustering is the computational task to partition a given input into subsets of equal characteristics. These subsets are usually called clusters and ideally consist of similar objects that are dissimilar to objects in other clusters. This way one can use clusters as a coarse representation of the data. We loose the accuracy of the original data set but we achieve simplification (this is somewhat comparable to lossy compression). When we deal with large data sets clustering the data may be the only possibility to visualize the structure of the data set as visualizing the whole set is typically not possible.
Clustering has many other applications in different areas of computer sciences such as computational biology, machine learning, data mining and pattern recognition. Since the quality of a partition is rather problem dependent, there is no general clustering algorithm. In this talk we consider k-means clustering, which is a widely used formulation of clustering. In this problem we are given a set P of n points in the Euclidean space $\mathbb{R}^d$. The goal is to find a set C of k points (called *centers*), such that

$$\sum_{p \in P} (d(p, C))^2$$

is minimized, where $d(p, C)$ denotes the distance of point p to the nearest point in C.
In this talk, we want to develop a simple coreset construction for k-means clustering [1]. A coreset is a small weighted points sets (point weights stand for multiplicities of points) such that for any set of k centers the cost of the coreset is within $(1 \pm \epsilon)$ times the cost of the original point set. The coreset we compute has size in $O(\log n)$ for constant ϵ and d. We present a dynamic data structure (e.g., one supporting insertions and deletions) that maintains in poly$(\log n)$ space such a coreset for a sequence of n insertions and deletion of points. Once we have computed such a coreset we can use an arbitrary algorithm to obtain a good clustering.
In the second part of this talk we show how to use these coresets to obtain a fast implementation of Lloyd's algorithm [4], which is one of the most widely used heuristic for k-means clustering and clustering in general [2]. We start with a small coreset and run a variant of this algorithm on it. Then we move to a coreset of bigger size and run our variant on it using the solution from the previous coreset as starting solution. We continue this process until our coreset equals the whole point set. The variant of Lloyd's algorithm we use is a variant of the KMHybrid algorithm [3, 5].

References

[1] G. Frahling and C. Sohler. Coresets in Dynamic Geometric Data Streams. *Proceedings of the 37th Annual ACM Symposium on Theory of Computing* (STOC'05), pp. 209–217, 2005.

[2] G. Frahling and C. Sohler. A fast k-means implementation using coresets. *to appear in Proceedings of the 22nd ACM Symposium on Computational Geometry* (SoCG'06), 2006.

[3] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu. An Efficient k-Means Clustering Algorithm: Analysis and Implementation. IEEE Trans. Pattern Anal. Mach. Intell. 24(7): 881-892, 2002.

[4] S. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28: 129–137, 1982.

[5] D. Mount. A Testbed for k-Means Clustering Algorithms. Available at http://www.cs.umd.edu/mount/Projects/KMeans/km-local-doc.pdf.

Learning in Games

Thomas Böhme

(joint work with Frank Göring, Jens Schreyer, Zsolt Tuza, Herwig Unger)

The results sketched in this talk are motivated by the following problem. Consider finitely many agents acting in an unknown environment. Each agent seeks to attain a certain goal. To this end it can choose from finitely many actions. The problem we are concerned with is to analyze whether the agents can learn (by trial and error) to choose the right actions to accomplish their respective goals. We concentrate on the special case that there is no direct communication between the agents, i.e. the only information an agent can use in the learning process is the list of actions applied in the past and the respective outcomes. We model this situation by a variation of repeated games (cf. [3]). We consider two different kinds of stage games: games played on directed graphs and a variation of games in strategic form.

Games on graphs Let G be a finite graph, x_0 a vertex of G (called the *terminal vertex*), and $I = \{1, \dots, n\}$ a set of players. For the sake of simplicity we restrict consideration in what follows to directed graphs without directed cycles. The vertex set of G is partitioned into $n+1$ classes $X_0, X_1, \dots, X_n$ where X_0 consists of all vertices with no out-going edges. (These vertices are called *terminal vertices.*) Each terminal vertex $t \in X_0$ is associated with a subset $W(t) \subseteq I$. At the beginning of a game a token is placed at the initial vertex x_0. If the token is at a vertex $u \in X_p$, player p chooses an edge uv starting at u and moves the token from u to the vertex v. (In this case we say that player p has *touched* the vertex u.) The game ends if and only if the token is moved to a terminal vertex. (The condition that G has no directed cycles ensures that every game ends after finitely many steps.) If the game ends at terminal vertex t, the players in $W(t)$ *win* the game and all other players *lose* it. A *strategy* for a player p is a function f from X_p into the edge set of G such that $f(x)$ is an edge starting at x for every vertex $x \in X_p$. A strategy f for player p is said a *winning strategy* if p wins every play in which he/she applies strategy f.

Consider a sequence of such games. We say that player p applies a *deterministic k-learning rule* in this sequence if her strategy s_p^i in the ith game is uniquely determined by his/her strategies in the preceding k games and the respective outcomes.

(The strategies of p in the first k plays are arbitrary.) We consider the following special deterministic 1-learning rule. For its definition we assume that for each vertex x the set of edges starting at x is stored as a *linear list* equipped with the *successor* operator σ_x. (If e is the last element in this list, then $\sigma_x(e) := \mathsf{NIL}$ is defined to be a dummy symbol.)

LNWP algorithm (Latest Not Winning Position). We say that a player $p \in P$ applies the *learning rule of Latest Losing Position* — LNWP, for short — if the strategy s_p^{i+1} of p in the $i+1$st game is determined by the preceding game as follows:

$$s_p^{i+1}(x) = \begin{cases} \sigma_x(s_p^i(x)) & \text{if } p \text{ does not win the } i\text{-th game and } x \text{ is the last vertex} \\ & \quad \text{with } \sigma_x(s_p^i(x)) \neq \mathsf{NIL} \text{ touched by } p \text{ in the } i\text{th game} \\ s_p^i(x) & \text{otherwise} \end{cases}$$

for each $x \in X_p$ and $i \geq 1$.

We mention, that if the learning rule LNWP is applied to a game with just one player p, then it visits the positions in the same order as depth-first search until the first terminal vertex t with $p \in W(t)$ (if there is any) has been found.
The following theorems are slightly simplified versions of results proved in [1].

Theorem 1. *Let $\gamma^1, \gamma^2, \ldots$ a sequence of games in which player p applies the learning rule LNWP, and let m denote the number of edges of G.*

(a) *If p has a winning strategy, then there are less than m games γ^i such that p does not win γ^i.*
(b) *If p does not win γ^i and $\sigma_x(s_p^i(x)) = \mathsf{NIL}$ for all x touched by p in the game γ^i, then p has no winning strategy.*

Theorem 1 does not imply that there is a number L (depending on G only) such that if player p has a winning strategy, then he/she will win every game γ^i after having played at least L games. For example, p may win the first 10,000 games without using a winning strategy, and none of the other players changes his/her strategy. Then, in the $10{,}001^{st}$ game some of the other players change their strategies and p loses this game. Our next theorem shows that there is such a bound, provided the other players also apply some kind of learning rule.

Theorem 2. *For every integer k there is an integer $L(k)$ (depending on the graph G) such that if player p has a winning strategy and $\gamma^1, \gamma^2, \ldots$ is a sequence of games in which p applies the learning rule LNWP and every other player applies some k-learning rule, then player p wins every game γ^i with $i > L(k)$.*

It is worth mentioning that neither Theorem 1 nor Theorem 2 guarantee that the player applying the learning rule LNWP will find a winning strategy. (A player

may win every game in a sequence without using a winning strategy just because the other players do not use all their strategies.)
We generalize the notion of a winning strategy as follows. Let $Q \subseteq I$ be a set of players. A family $\{f_q \mid q \in Q\}$ of strategies is called a *collaborative winning strategy* for Q if each player in Q will win every game in which **every** player $p \in Q$ apply the strategy f_q. There are examples of games (played on a suitable directed graph) with three players $\{1, 2, 3\}$ such that no player has a winning strategy, every 2-element set of players has a collaborative winning strategy, and the set $\{1, 2, 3\}$ has no collaborative winning strategy. Furthermore, even if all players apply the learning rule LNWP in an infinite sequence of such games, it may happen that every player loses infinitely many games. This motivates our second result.

Strategic-Form Games A *game in strategic form* (cf. [3]) consists of a set of players $I = \{1, \ldots, n\}$, a family $\{S_i \mid i \in I\}$ of strategy sets, and a family $\{u_i \mid i \in I\}$ of pay-off functions. Each pay-off function u_i is a real-valued function defined on $S_1 \times \cdots \times S_n$. We restrict consideration to strategic-form games with the property that all pay-off functions u_i take their values in $\{0, 1\}$. (Henceforth, such games are called *0-1-games.*) In case of a 0-1-game, we say that player i *wins* the game if $u_i = 1$ and *loses* it if $u_i = 0$. Adopting this notation, we can apply the notions k-learning rule and collaborative winning strategy defined in the previous section analogously to 0-1-games. We also say that a player applies a *randomized k-learning rule* in a sequence of games if the distribution of his/her strategy in the ith game is uniquely determined by the his/her strategies in the preceding k games and the respective outcomes.
Consider a sequence $\gamma^1, \gamma^2, \ldots$ of such games, and denote the strategy of player p in the ith game by s^i_p. Suppose that every player applies the following randomized 2-learning rule. If $s^{i-1}_p = s^{i-2}_p$ and p wins the $(i-2)$nd and the $(i-1)$st game, p keeps his/her strategy in the ith game, i.e. $s^i_p = s^{i-1}_p = s^{i-2}_p$. In all other cases p chooses his/her strategy s^i_p in the ith game uniformly at random. It is proved in [2] that if some subset of players has a collaborative winning strategy and each strategy set S_i has at least three elements, then the probability that for some subset $Q' \subseteq I$ of players all players $q \in Q'$ apply a collaborative winning strategy in all games γ^i with $i > n$ converges to 1 with n tending to ∞. A closely related result about almost sure convergence to Nash equilibria was proved by S. Hart and A. Mas-Colell [4]. It also follows from a result in [4] that no randomized 1-learning rule guarantees almost sure convergence to a collaborative winning strategy (see also [2]).

References

[1] T. Böhme, F. Göring, Zs. Tuza, and H. Unger *Learning of Winning Strategies for Terminal Games with Linear-Size Memory*, in preparation.
[2] T. Böhme and J. Schreyer *Learning to cooperate with almost no communication*, submitted to ICML-2006.
[3] D. Fudenberg and J. Tirole, *Game Theory*, The MIT Press Cambridge, Massachustes; London, England (1993).

[4] S. Hart and A. Mas-Colell, *Stochastic uncoupled dynamics and Nash equilibrium*, Technical report, The Hebrew University of Jerusalem, 2005

Reporter: Stephan Matos Camacho

Participants

Prof. Dr. Evripidis Bampis
IBISC
CNRS FRE 2873
Universite d'Evry Val-d'Essone
F-91025 Evry Cedex

Dr. Daniel Bienstock
Dept. of Industrial Engineering and
Operations Research, Columbia Univ.
Seeley W., Mudd Building
500 West 120th Street
New York, NY 10027
USA

Dr. Hans Bodlaender
Department of Computer Science
Utrecht University
P. O. Box 80. 089
Padualaan 14
NL-3508 TB Utrecht

Dr. Thomas Böhme
Institut für Mathematik
Technische Universität Ilmenau
Helmholtzplatz 1
98693 Ilmenau

Prof. Dr. Andreas Brandstädt
Institut für Informatik
Universität Rostock
18051 Rostock

Dr. Stephan Brandt
Institut f. Mathematik
Technische Universität Ilmenau
Weimarer Str. 25
98693 Ilmenau

Prof. Dr. Hajo Broersma
Dept. of Computer Science
University of Durham
South Road
GB-Durham DH1 3LE

Florian Diedrich
Institut für Informatik und
Praktische Mathematik
Christian-Albrechts-Universität
Olshausenstr. 40
24098 Kiel

Prof. Dr. Martin E. Dyer
The School of Computing
University of Leeds
GB-Leeds LS2 9JT

Dr. Frank Göring
Fakultät für Mathematik
Technische Universität Chemnitz
Reichenhainer Str. 39
09126 Chemnitz

Prof. Dr. Mirko Hornak
P.J. Safarik University
Institute of Mathematical Sciences
Jesenna 5
040 01 Kosice
Slovakia

Dr. Stefan Hougardy
Institut für Informatik
HU-Berlin
Unter den Linden 6
10099 Berlin

Prof. Dr. Klaus Jansen
Institut für Informatik
Universität Kiel
Olshausenstr. 40
24118 Kiel

Prof. Dr. Arnfried Kemnitz
Institut Computational Mathematics
Technische Universität Braunschweig
Pockelstraße 14
38106 Braunschweig

Prof. Dr. Lefteris M. Kirousis
University of Patras
Computer Eng. & Informatics Dept.
University Campus
26504 Patras
GREECE

Prof. Dr. Jochen Könemann
Dept. of Combinatorics and
Optimization
University of Waterloo
200 University Avenue West
Waterloo, Ont. N2L 3G1
CANADA

Dipl.Math. Anja Kohl
Fak. für Mathematik und Informatik
Bergakademie Freiberg
09596 Freiberg

Prof. Dr. Dieter Kratsch
Universite de Metz
LITA
Ile du Saulcy
F-57045 Metz Cedex 01

Dr. Van Bang Le
Institut für Theoretische
Informatik, FB Informatik
Universität Rostock
Albert-Einstein-Straße 21
18051 Rostock

Prof. Dr. Tomasz Luczak
Zaklad Matematyki Dyskretnej
Wydzial Matematyki i Informatyki
Uniwersytet im. Adama Mickiewicza
61-614 Poznan
POLEN

Stephan Matos Camacho
Fakultät für Mathematik und
Informatik; Technische Universität
Bergakademie Freiberg
Prüferstr. 1
09599 Freiberg

Prof. Dr. Friedhelm Meyer auf der Heide
Heinz-Nixdorf Institut &
Institut für Informatik
Universität Paderborn
Fürstenallee 11
33102 Paderborn

Prof. Dr. Rolf Möhring
Institut für Mathematik - Fak. II
Technische Universität Berlin
Sekr. MA 6-1
Straße des 17. Juni 136
10623 Berlin

Dr. Alantha Newman
Max-Planck-Institut für Informatik
Geb. 46
Stuhlsatzenhausweg 85
66123 Saarbrücken

Prof. Dr. Harald Räcke
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213-3890
USA

Dr. Bert Randerath
Institut für Informatik
Universität zu Köln
Pohligstr. 1
50969 Köln

Prof. Dr. Dieter Rautenbach
Forschungsinstitut für
Diskrete Mathematik
Universität Bonn
Lennestr. 2
53113 Bonn

Prof. Dr. Zdenek Ryjacek
Department of Mathematics
University of West Bohemia
Univerzitni 22
306 14 Pilsen
Czech Republic

Dr. Piotr Sankowski
Institute of Informatics
Warsaw University
ul.Banacha 2
02-097 Warsaw
Poland

Prof. Dr. Christian Scheideler
Department of Computer Science
Johns Hopkins University
3400 N. Charles Street
Baltimore, MD 21218-2682
USA

Prof. Dr. Ingo Schiermeyer
Fakultät für Mathematik und
Informatik; Technische Universität
Bergakademie Freiberg
Prüferstr. 1
09599 Freiberg

Dr. Christian Schindelhauer
Heinz-Nixdorf Institut &
Institut für Informatik
Universität Paderborn
Fürstenallee 11
33102 Paderborn

Ulrich Michael Schwarz
Institut für Informatik
Universität Kiel
Olshausenstr. 40
24118 Kiel

Christian Sohler
Heinz-Nixdorf Institut &
Institut für Informatik
Universität Paderborn
Fürstenallee 11
33102 Paderborn

Prof. Dr. Martin Sonntag
Fakultät für Mathematik und
Informatik; Technische Universität
Bergakademie Freiberg
Prüferstr. 1
09599 Freiberg

Prof. Dr. Paul Spirakis
Research & Academic Computer
Technology Institute (RACTI)
N. Kazantzaki Str.
P.O.Box 1122, Rion
265 00 Patras
GREECE

Prof. Dr. Endre Szemeredi
Department of Computer Science
Rutgers University
Hill Center, Busch Campus
New Brunswick, NJ 08903
USA

Dr. Hanns-Martin Teichert
Institut für Mathematik
Universität Lübeck
Wallstr. 40
23560 Lübeck

Prof. Dr. Peter Tittmann
Fachbereich Mathematik, Physik und Informatik
Hochschule Mittweida
Technikumplatz 17
09648 Mittweida

Prof. Dr. Berthold Vöcking
Lehrstuhl für Informatik I
RWTH Aachen
Ahornstr. 55
52074 Aachen

Dr. Margit Voigt
Fachbereich Informatik
Hochschule für Technik und Wirtschaft
Friedrich-List-Platz 1
01069 Dresden

Dr. Annegret Wagler
Institut für Mathematische Optimierung
Universität Magdeburg
Universitätsplatz 2
39106 Magdeburg

Prof. Dr. Emo Welzl
Theoretische Informatik
ETH-Zürich
ETH-Zentrum
CH-8092 Zürich

Prof. Dr. Peter Widmayer
Institut für Informatik
ETH-Zürich
ETH-Zentrum
CH-8092 Zürich

Prof. Dr. Gerhard Woeginger
Department of Mathematics
Eindhoven University of Technology
P.O.Box 513
NL-5600 MB Eindhoven

Prof. Dr. Mariusz Wozniak
Department of Applied Mathematics
AGH University of Science and Technology
Al. Mickiewicza 30
30-059 Krakow
POLAND