Enineering Algorithms for Approximate Weighted Matching[1]

Jens Maue[1] Peter Sanders[2]

[1]ETH Zürich, Switzerland

[2]Universität Karlsruhe (TH), Germany

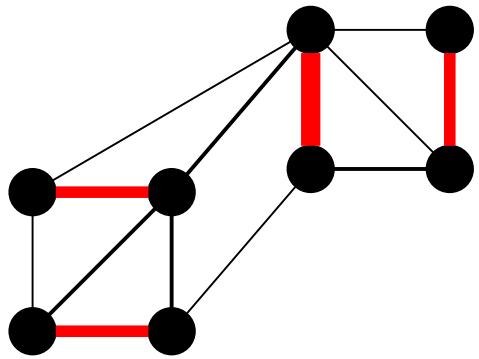
Part of this work was done at Max-Planck-Institut für Informatik, Saarbrücken, Germany.

07 June 2007

[1]Partially supported by DFG grants SA 933/1-2, SA 933/1-3.

Introduction

Given: weighted graph $G = (V, E)$.
Find max. weight $M \subseteq E$ s.t. (V, M) has max. degree 1.
Exact algorithm takes time $\mathcal{O}(n(m + n \log n))$ [Gabow 90].
Too slow for massive graphs.
E.g. graph contraction.
$\rightsquigarrow$ need for fast approximation algorithms.



Previous Approximation Algorithms

$\frac{1}{2}$-Approximations

- greedy algorithm $\mathcal{O}(m + \text{sort}(m))$ [Avis, 1983]
- first linear time: Preis $\mathcal{O}(m)$ [Preis, 1999]
- Path Growing Algorithm (PGA) $\mathcal{O}(m)$ [Drake and Hougardy, 2003b]
- improved by path heuristics (PGA') [Drake and Hougardy, 2003a]

$(\frac{2}{3} - \epsilon)$-Approximations

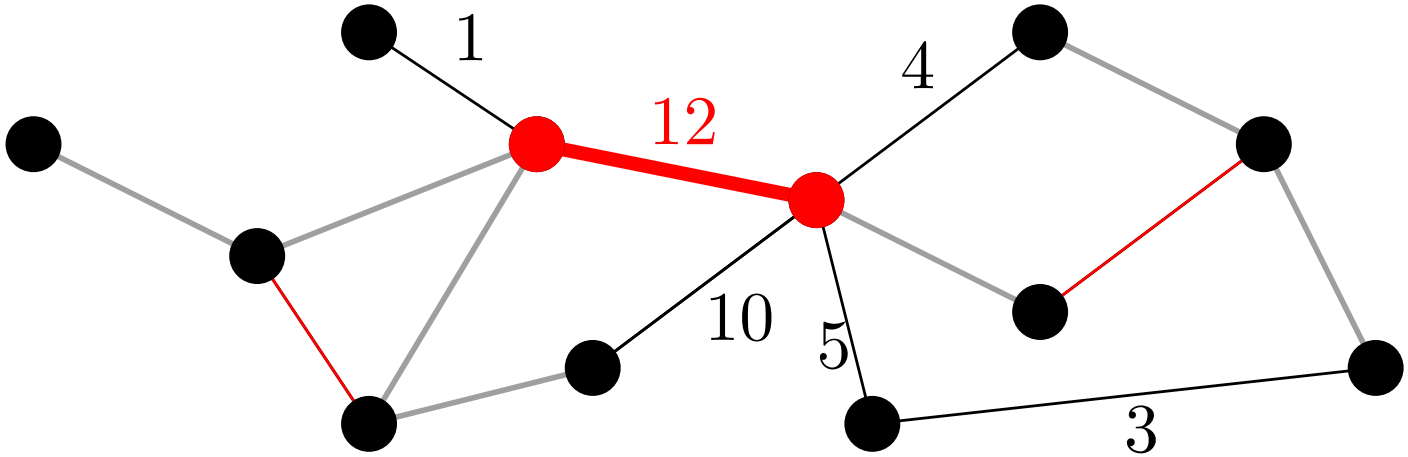
- $\mathcal{O}(\frac{n}{\epsilon})$ deterministic but complicated [Drake and Hougardy, 2003a] [Drake Vinkemeier and Hougardy, 2005]
- $\mathcal{O}(n \log \frac{1}{\epsilon})$ deterministic but complicated [Pettie and Sanders, 2004]
- $\mathcal{O}(n \log \frac{1}{\epsilon})$ expected. Simple (RAMA) [Pettie and Sanders, 2004]

Previous Experiments [Drake Hougardy 2003a]

Very interesting results. But

- No linear time 2/3 approximations (not quite)
- Limited instance sizes
- No real world instances
- Greedy fares worst

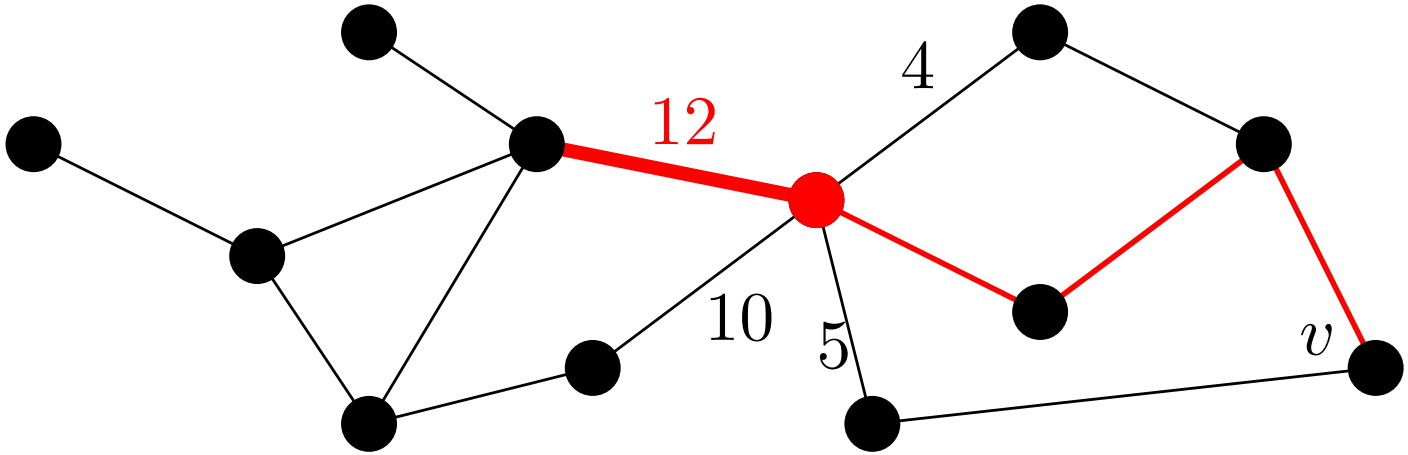
Greedy Algorithm [Avis 1983]



- Sort edges according to weight.
- Repeatedly add globally heaviest edge e.
- Remove edges adjacent to e.
- Running time $\mathcal{O}(m + \text{sort}(m))$

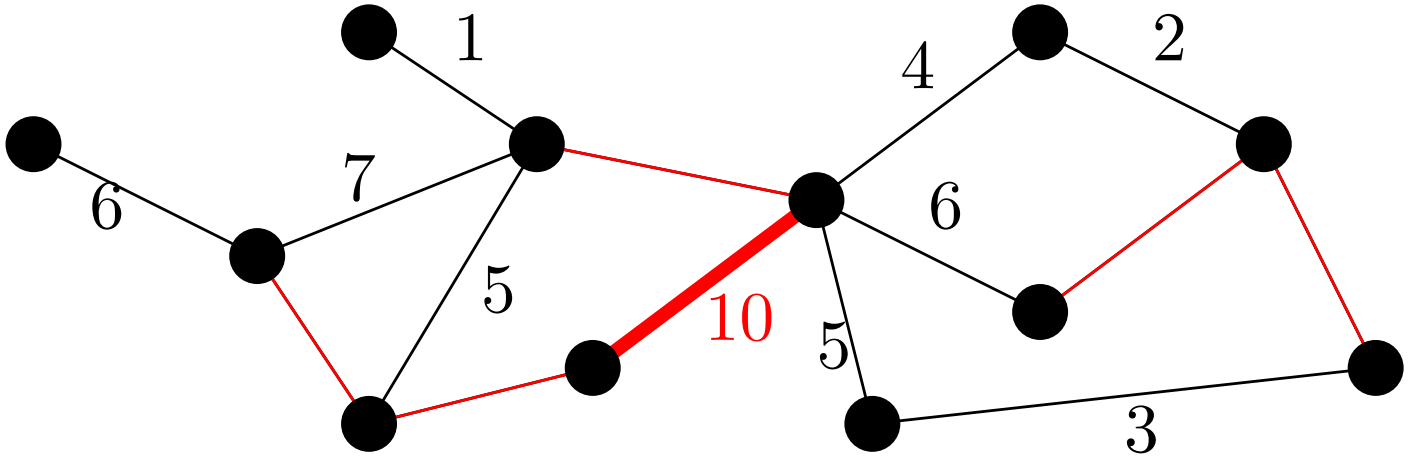
Observation: linear time for polynomial integer weights new ?

Path Growing Algorithm PGA' [Drake Hougardy 2003a]



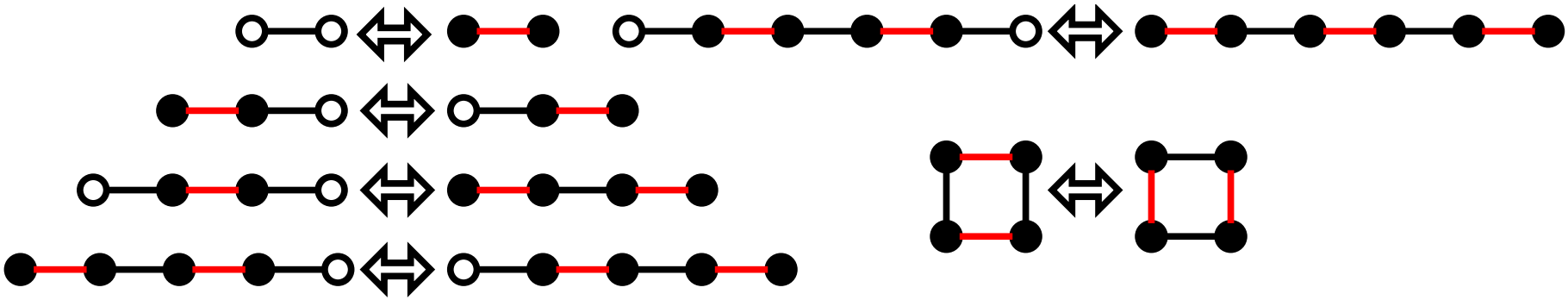
- Grow a set of paths:
  1. Start at arbitrary vertex v.
  2. Extend path by locally heaviest edge.
  3. If current path cannot be extended, start new path.
- Find optimum matching of every path in linear time. (dynamic programming)

Global Paths Algorithm GPA NEW



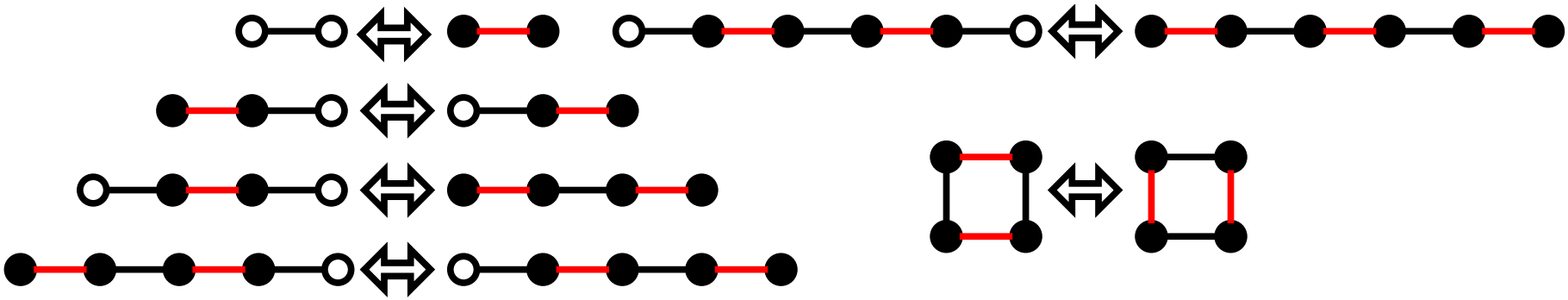
- Sort edges according to their weight.
- Grow a set of paths and even-length cycles:
  Repeatedly extend paths by globally heaviest applicable edge.
- Find optimum matching for every path and cycle.
- Running time $\mathcal{O}(m + \text{sort}(m))$ or less ...
- Approximation $\frac{1}{2}\text{OPT}$

Randomized $\frac{2}{3} - \epsilon$ Approximation Algorithm



- Find short augmenting paths/cycles centered at some vertex v.
- Apply augmentation of highest-gain.
- Random Augmentation Matching Algorithm (RAMA): repeatedly select a center node randomly.
- Expected running time $\mathcal{O}(m \log \frac{1}{\epsilon})$

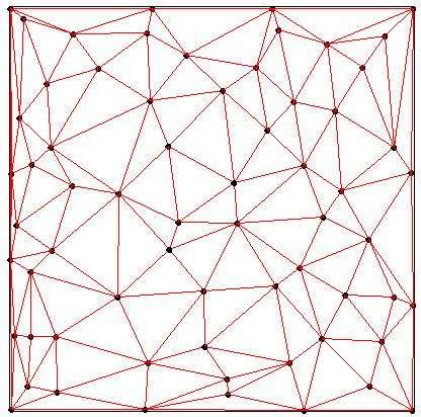
Randomized $\frac{2}{3} - \epsilon$ Approximation Algorithms



- Find short augmenting paths/cycles centered at some vertex v.
- Apply augmentation of highest-gain.
- Random Augmentation Matching Algorithm (RAMA): repeatedly select a center node randomly.
- Expected running time $\mathcal{O}(m \log \frac{1}{\epsilon})$
- Random Order Matching Algorithm (ROMA): repeatedly select all nodes in random order.

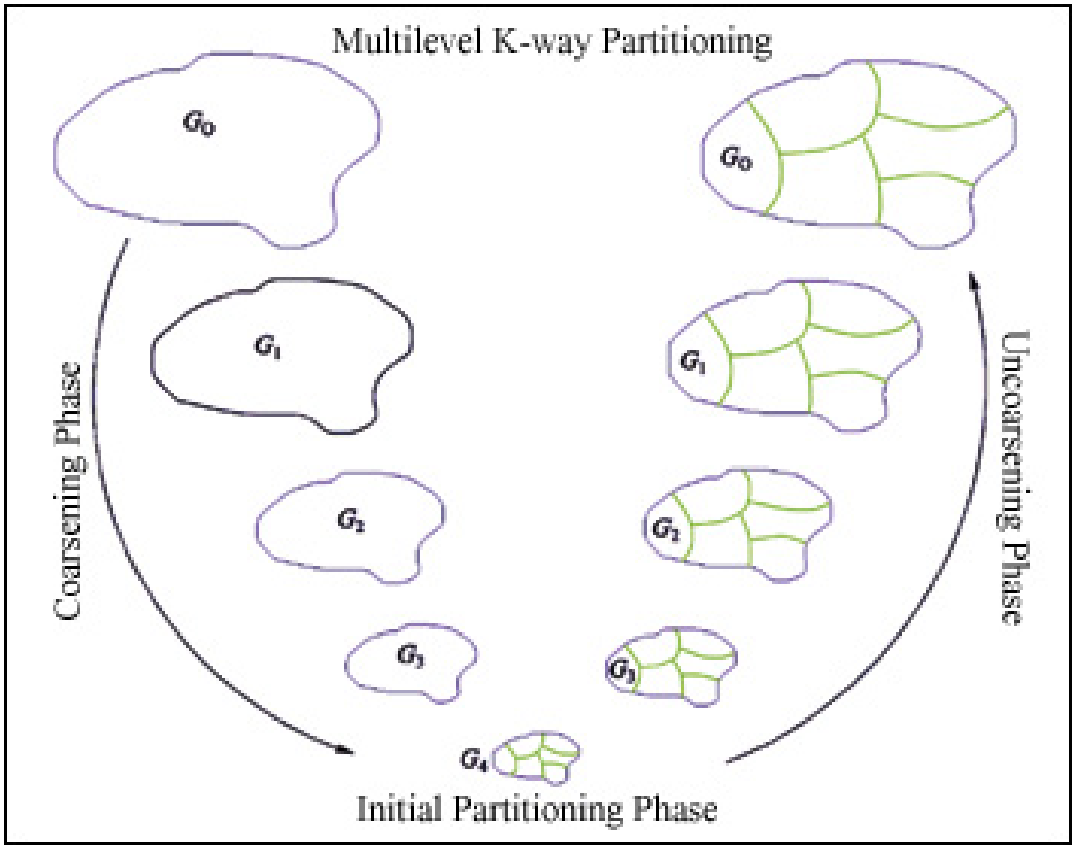
Synthetic Instances

- Random graphs
  $n = 2^{10} \ldots 2^{17}$, $\alpha := m/n = 2 \ldots 2^8$
- Complete geometric instances
  $n = 2^6 \ldots 2^{12}$ random points from a square grid
- Delaunay triangulations
  $n = 2^{10} \ldots 2^{18}$ random points for a square grid

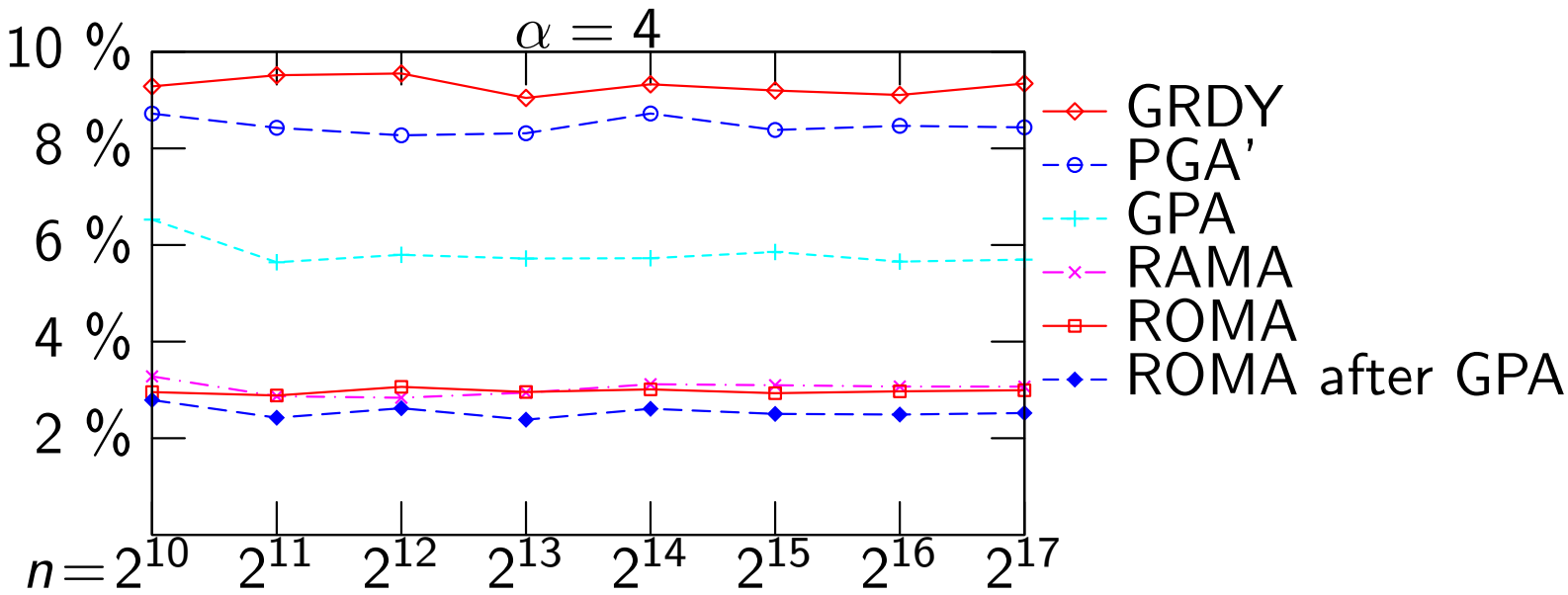


Real World Graph Partitioning Instance [Walshaw]

- Iteratively contract graphs (up to 8 $\times$)
- Weights = edge multiplicities (small integers !)
- n =42–448 695
  m =83–3 314 611

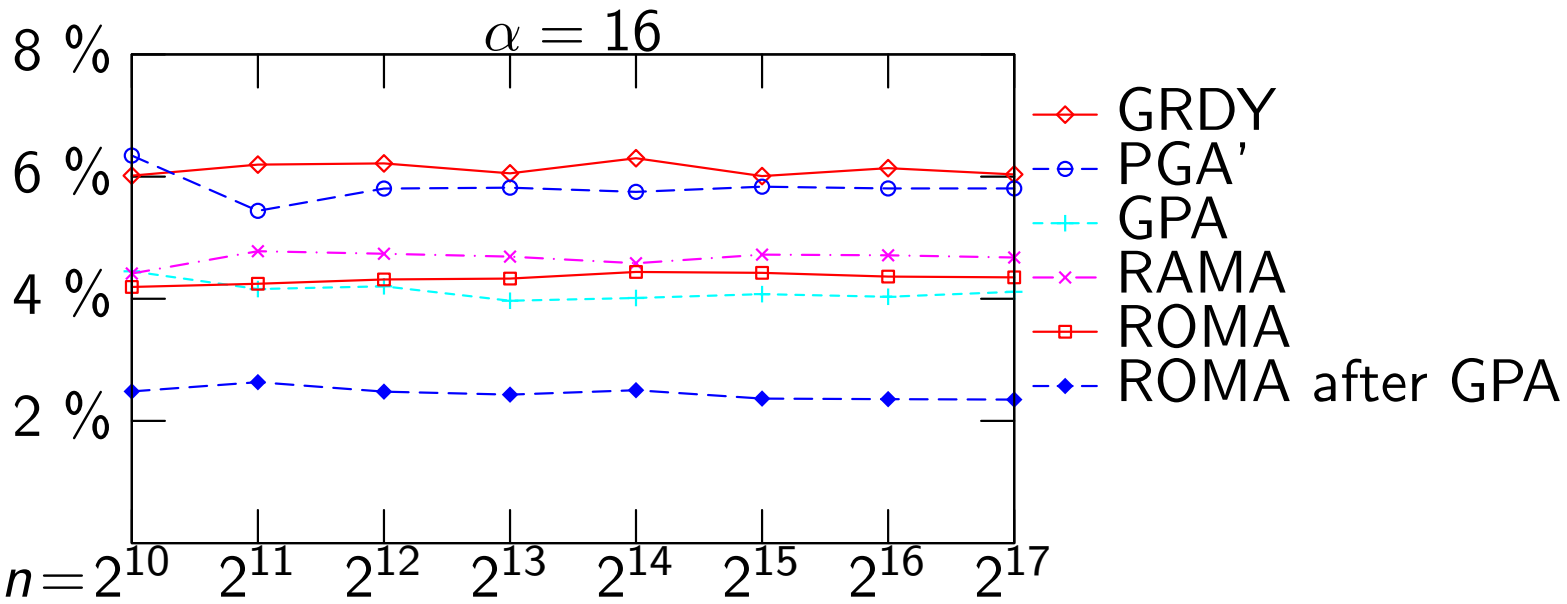


Random Graphs $\alpha := \frac{m}{n}$



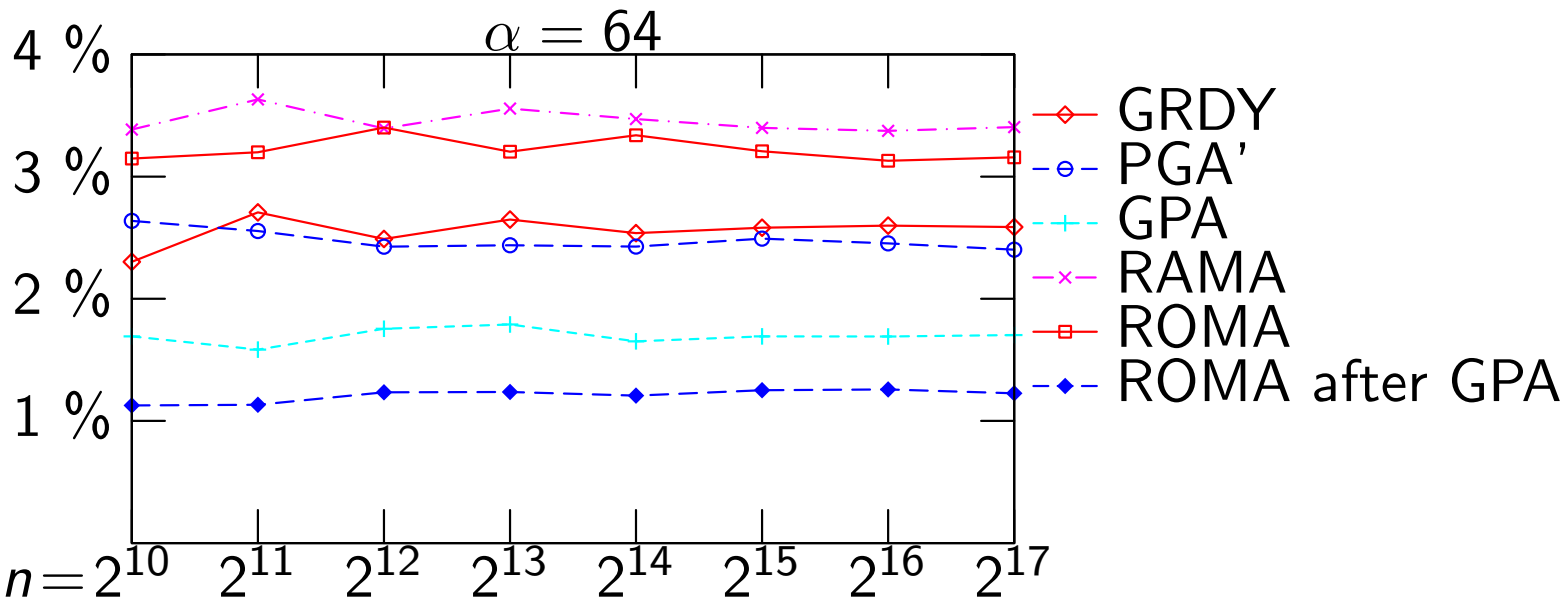
- GPA: always better than GRDY and PGA'.
- RAMA/ROMA:
  - better than GPA for small α...

Random Graphs $\alpha := \frac{m}{n}$



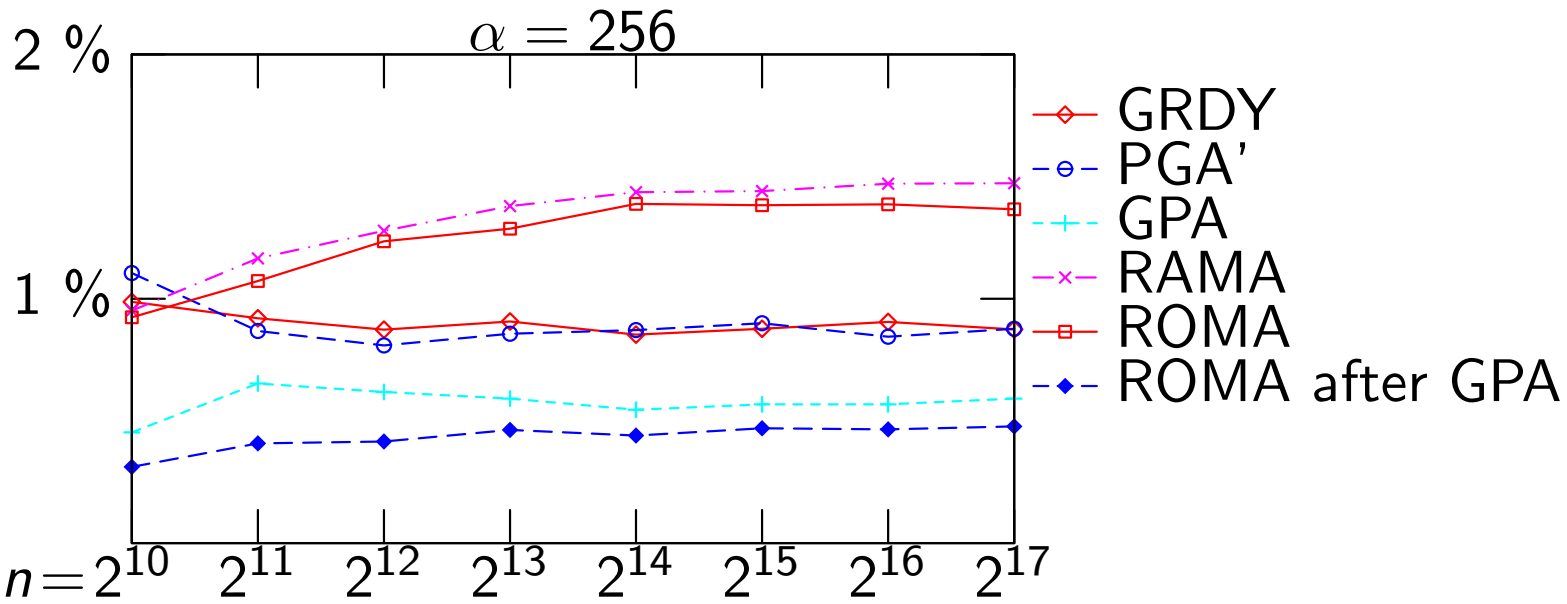
- GPA: always better than GRDY and PGA'.
- RAMA/ROMA:
  - better than GPA for small α...
  - equal to GPA for $\alpha = 16$...

Random Graphs $\alpha := \frac{m}{n}$



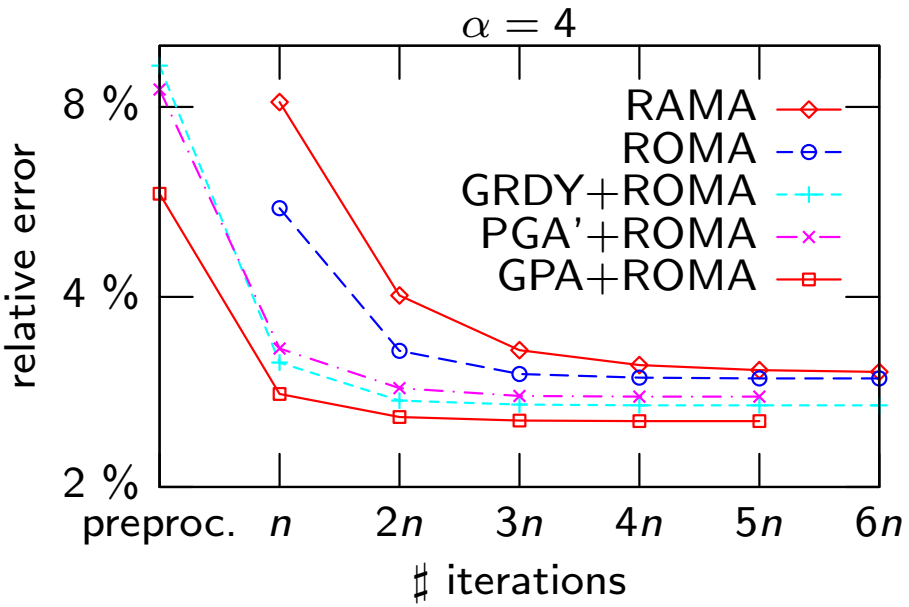
- GPA: always better than GRDY and PGA'.
- RAMA/ROMA:
  - better than GPA for small α...
  - equal to GPA for $\alpha = 16$...
  - worse than GPA for $\alpha > 16$...

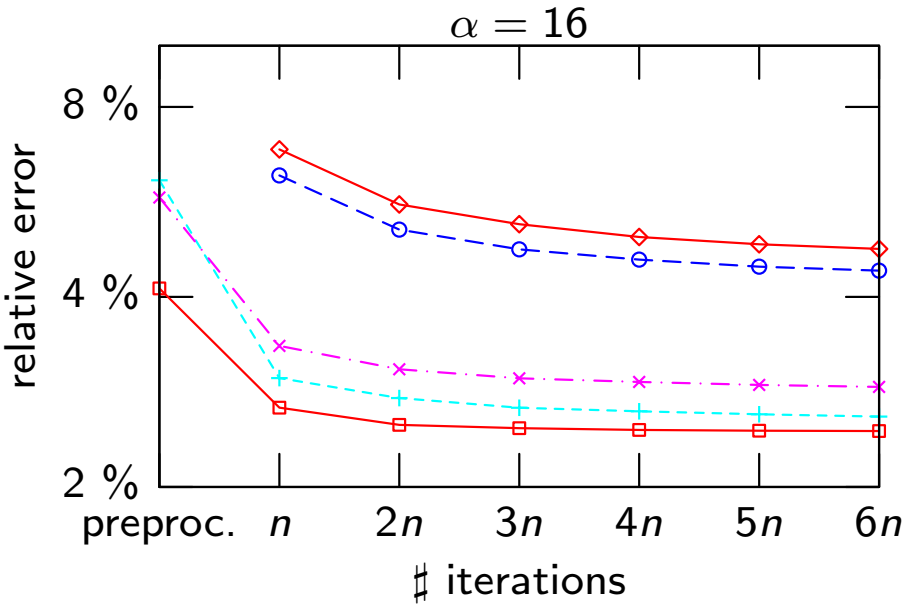
Random Graphs $\alpha := \frac{m}{n}$



- GPA: always better than GRDY and PGA'.
- RAMA/ROMA:
  - better than GPA for small α...
  - equal to GPA for $\alpha = 16$...
  - worse than GPA for $\alpha > 16$...
  - worse than GRDY and PGA for bigger α.
  - ROMA with initial GPA: always best.

Convergence of Randomized Algorithms



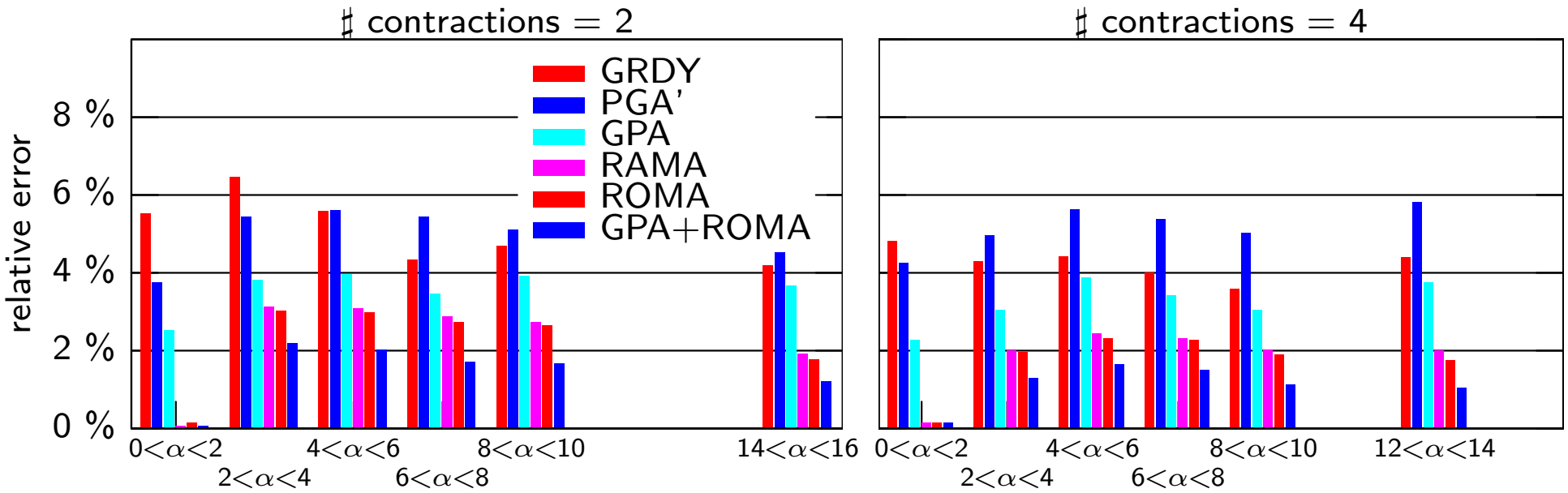


Convergence of randomized algorithms

- stand-alone: ROMA better than RAMA
- improvement for any initialization method
- best: preprocessing with GPA

Delaunay, Complete Geometric, Real World

Similar results as for random graphs with same density



Running Time

- Implementations not highly tuned (except OPT (LEDA))
  $\rightsquigarrow$ results are preliminary
- $T(\mathrm{PGA}') \leq T(\mathrm{GPA})$ but considerably worse quality
- $T(\mathrm{GPA}) \leq T(\mathrm{ROMA})$!
  log factors in sorting are often irrelevant compared to cache faults,. . .
- $T(\mathrm{GPA} + \mathrm{ROMA}) \approx T(\mathrm{ROMA})$!
  (for same quality)
- $T(\mathrm{OPT}) \ll$ worst case

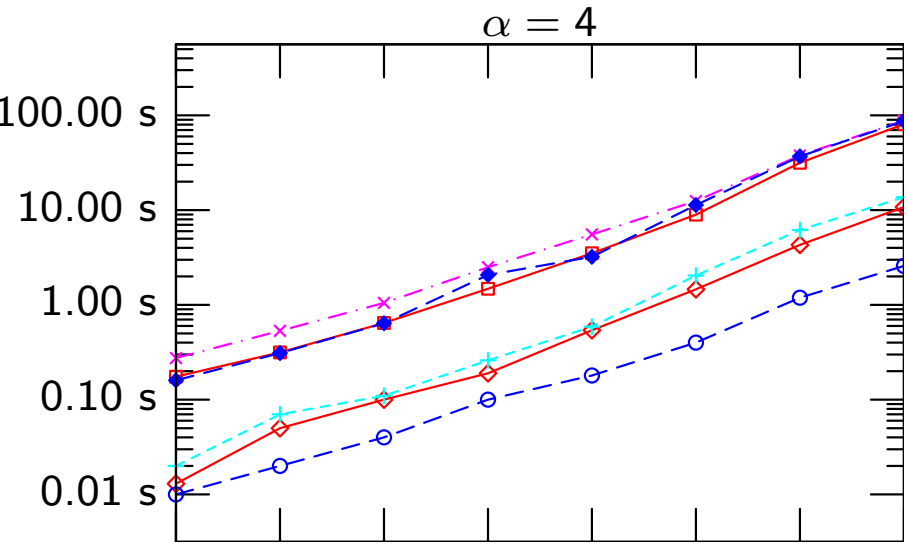
Conclusions

- All algs within a few percent of optimum even for real world inputs
- GPA combines advantages of greedy and PGA'.
  Sorting overhead has been overemphasized for
  both theoretical and practical reasons
- ROMA improves on RAMA
- GPA+ROMA improves on ROMA

Conclusions and Future Work

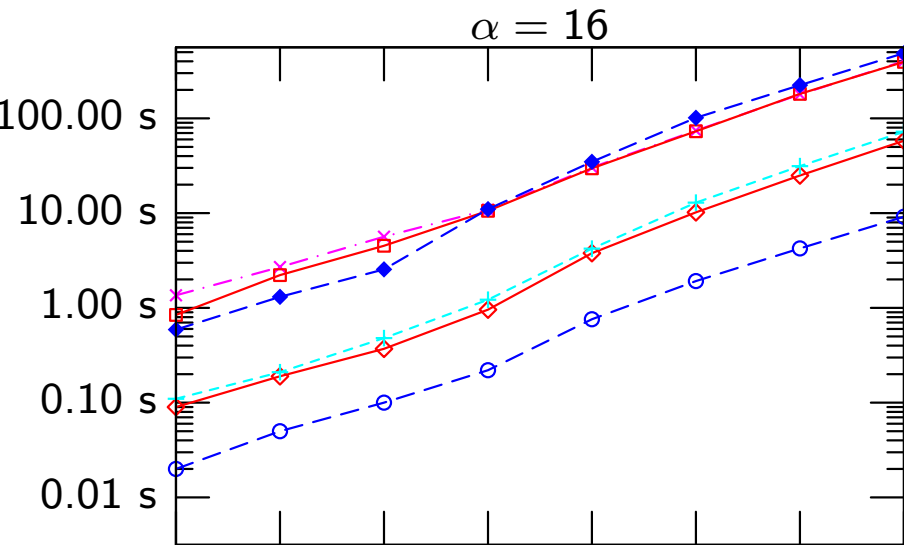
- All algs within a few percent of optimum even for real world inputs
- GPA combines advantages of greedy and PGA'.
  Sorting overhead has been overemphasized for
  both theoretical and practical reasons
- ROMA improves on RAMA
- GPA+ROMA improves on ROMA
- Prove that ROMA works also in theory
- Tuned implementation of GPA+ROMA.
  cache eff. (integer) sorting, graph representation, path-matching,...
- Randomized algorithms:
  maintain pool of candidates for augmentation
- Parallelize sorting, path matching,...

Random Graphs



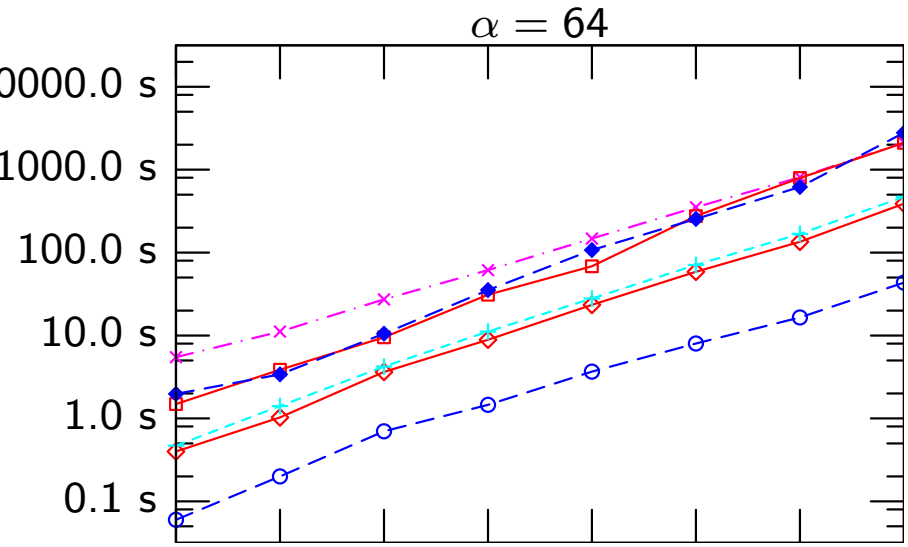
- edge-node ratio $\alpha = \frac{m}{n}$
- PGA': always fastest
- superlinear algorithms also reasonable in practice

Random Graphs



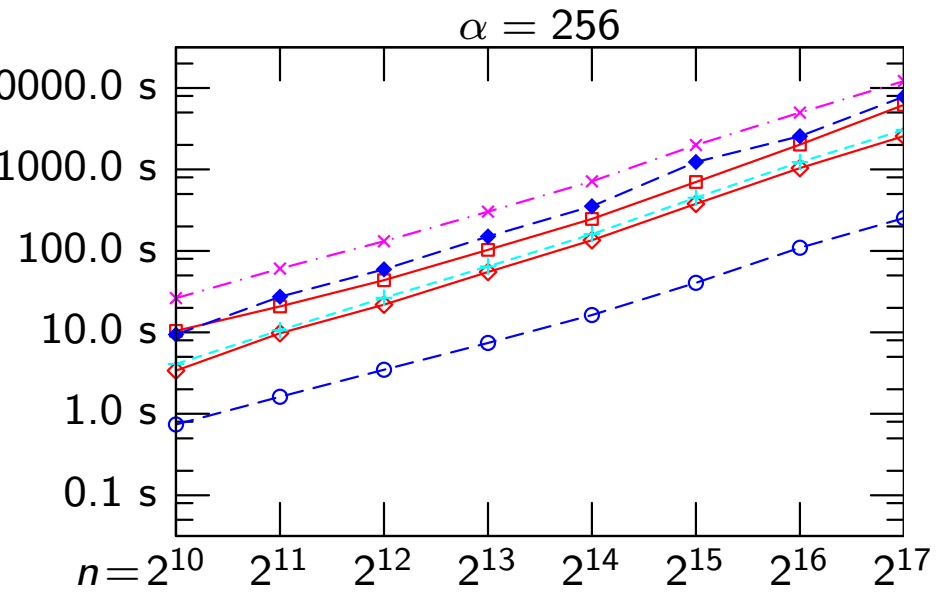
- edge-node ratio $\alpha = \frac{m}{n}$
- PGA': always fastest
- superlinear algorithms also reasonable in practice

Random Graphs



- edge-node ratio $\alpha = \frac{m}{n}$
- PGA': always fastest
- superlinear algorithms also reasonable in practice

Random Graphs



- edge-node ratio $\alpha = \frac{m}{n}$
- PGA': always fastest
- superlinear algorithms also reasonable in practice


Avis, D. (1983).
A survey of heuristics for the weighted matching problem.
Networks, 13(4):475–493.


Drake, D. E. and Hougardy, S. (2003a).
Linear time local improvements for weighted matchings in graphs.
In *Proceedings of the 2nd International Workshop on Experimental and Efficient Algorithms (WEA-03)*, volume 2647 of *LNCS*, pages 107–119. Springer.


Drake, D. E. and Hougardy, S. (2003b).
A simple approximation algorithm for the weighted matching problem.
Information Processing Letters, 85(4):211–213.


Drake Vinkemeier, D. E. and Hougardy, S. (2005).
A linear-time approximation algorithm for weighted matchings in graphs.
ACM Trans. Algorithms, 1(1):107–122.


Pettie, S. and Sanders, P. (2004).

A simpler linear time $2/3 - \epsilon$ approximation for maximum weight matching.
Information Processing Letters, 91(6):271–276.

Preis, R. (1999).
Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs.
In *Proc. of the 16th Annual Symp. on Theoretical Aspects of Computer Science (STACS-99)*, volume 1563 of *LNCS*, pages 259–269. Springer.

Walshaw, C.
The graph partitioning archive.
`http://staffweb.cms.gre.ac.uk/~c.walshaw/partition/`.