THE UNIVERSITY OF ADELAIDE
AUSTRALIA

Features of Easy and Hard Instances for Approximation Algorithms and the Traveling Salesperson Problem

Markus Wagner

TU Dortmund University
Bernd Bischl, Olaf Mersmann, and Heike Trautmann
The University of Adelaide
Samadhi Nallaperuma, Frank Neumann

Features of Easy and Hard Instances for Approximation Algorithms and the Traveling Salesperson Problem

or

"What makes TSP instances difficult?"

Markus Wagner

TU Dortmund University
Bernd Bischl, Olaf Mersmann, and Heike Trautmann
The University of Adelaide
Samadhi Nallaperuma, Frank Neumann

Summary

- Understanding the behavior of TSP solvers is still difficult.

- Here:
  - Three algorithms
    - 2-Approximation
    - 3/2-Approximation (Christofides)
    - Local search 2-opt*
  - Feature-based comparison
    - what makes instances hard/easy
  - Cross-comparison

School of Computer Science

Motivation, and the Travelling Salesperson Problem (TSP)

http://blogs.oc.edu/ee/images/uploads/ece/italyTour2_thumb.png

Motivation

- Understanding algorithm performance for hard optimization tasks is still difficult
- More precisely: given an instance I, it is often hard to predict the performance of an algorithm A, without running A on I.
- Classical approach (worst/average case) hardly captures real-world performance
- Hyper heuristics (optimization domain, machine learning) focus on finding the conditions that determine the algorithm performance in advance.
- Understanding important for (automated) algorithm selection.

Motivation

- Smith-Miles, Lopez [8] classify directions
  - Automatic algorithm selection is based on (learned knowledge from) previous algorithm performance
  - Analyze algorithms and problems theoretically/experimentally, to understand the reasons, to influence future algorithm design for more complex problems
- Here: we do both
  - We generate hard/easy instances, and characterize them
  - Insights can be used for performance prediction to support algorithm selection.

Travelling Salesperson Problem (TSP)

- Famous combinatorial NP-hard problem
- Given a set of *n* cities {1,…,n}, and a distance matrix $d=(d_{ij})$, $1 \leq i,j \leq n$, the task is to compute a tour of minimal length, which visits each cities once, and returns to the origin.
- Euclidian TSP (cities in the plane, Euclidian distances) is still NP-hard, and a special case of the Metric TSP (distances fulfill triangle inequalities, different approximation algorithms are known)

THE UNIVERSITY OF ADELAIDE AUSTRALIA

Instance Generation and Investigated Features

Instance Generation

Approach by Mersmann et al. [6]

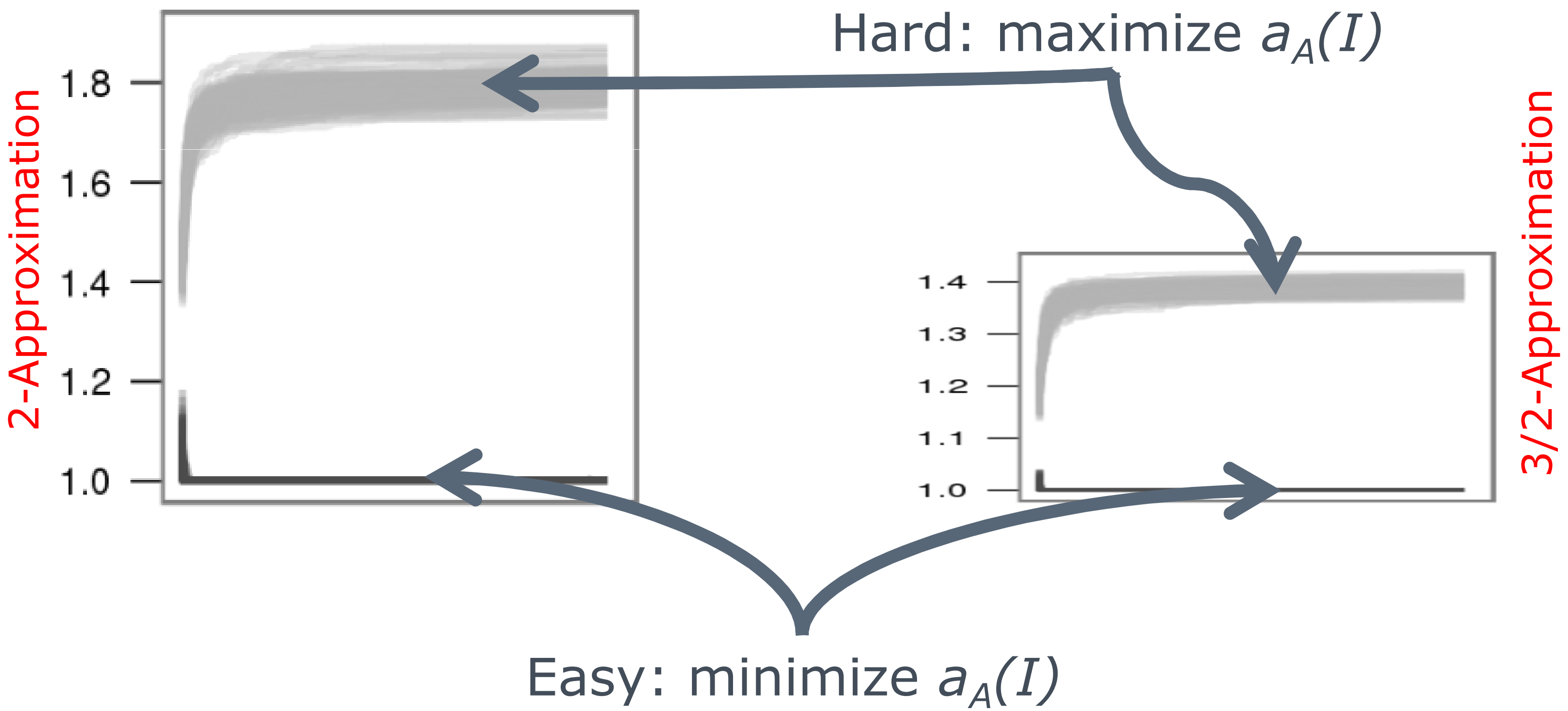
- Evolve hard/easy instances with an evolutionary algorithms
- Difficulty assessment (other measures possible)
  - $a_A(I)$ approximation ratio of algorithm A on instance I
  - $a_A(I)=A(I)/OPT(I)$

Note:

- an algorithm is a r-approximation algorithm,
  if $a_A(I) \leq r$ holds for all instances.
- $OPT(I)$ computed by CONCORDE [1]

Instance Generation

Use non-determinism: repeat runs for collections



Investigated Features

Mersmann et al. [6] (total: 46 features)

- Distance features: the cost distribution …
- Cluster features: number of cluster, mean distance to centroid, …
- Nearest Neighbour distance features: minimum, maximum, mean, standard deviation, …
- Centroid features: coordinates, distance to other nodes, …
- MST features: min/max/stdev/… of depth/distance values,…
- Angle features (between node and its two NN): min/max/…
- Convex Hull features: area, number of nodes defining CH, …

THE UNIVERSITY OF ADELAIDE AUSTRALIA

Analysis: 2-Approximation

3/2-Approximation

Algorithm 1: 2-Approximation algorithm

input : Graph G

output: Hamiltonian cycle S

Subgraph connecting all vertices, with minimum weight

Build a minimum spanning tree MST T of G;

Duplicate all edges, forming the graph U;

Create an Euler cycle S in U;

Make S Hamiltonian by skipping already seen nodes;

return S;

improvement

Algorithm 2: Christofides 3/2-approximation algorithm

Set of pair-wise non-adjacent edges

⋮

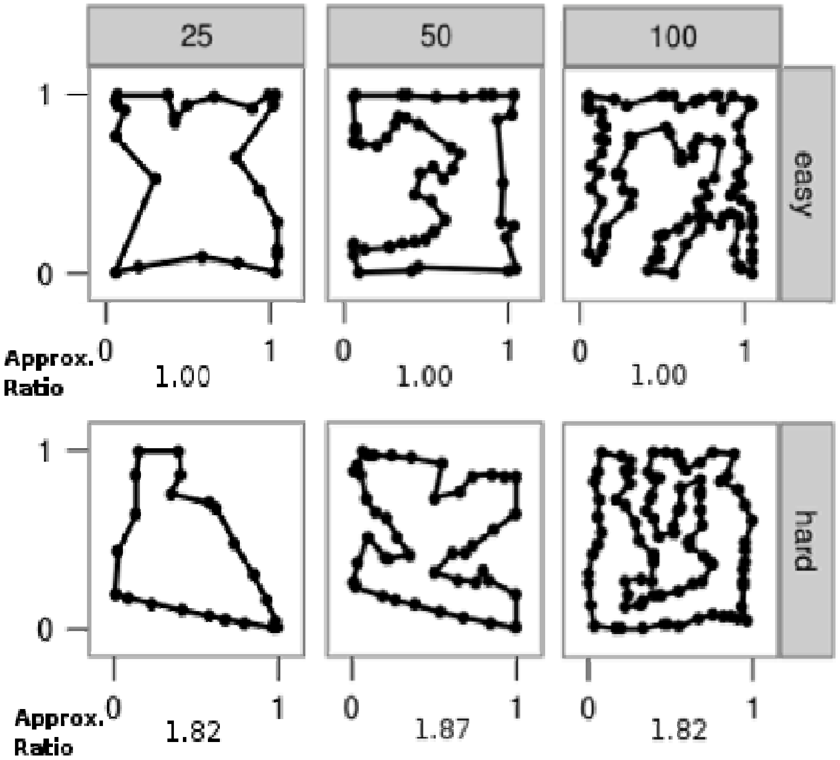
Find a minimum-weight perfect matching M on the set of nodes having an odd degree;

Combine the edges of M and T to form the graph U;

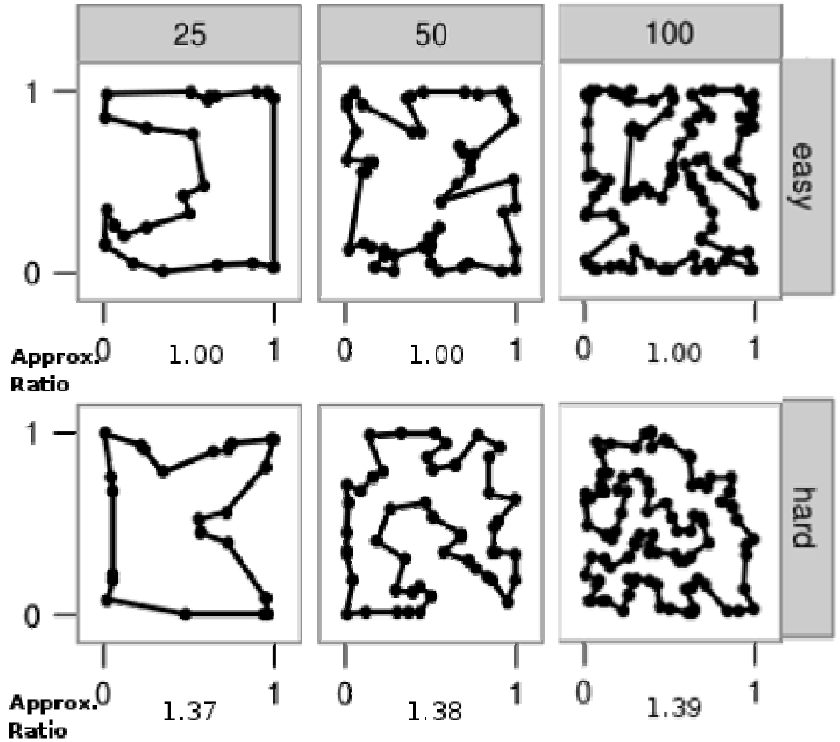
⋮

Step 1: Eye-Balling

2-Approximation

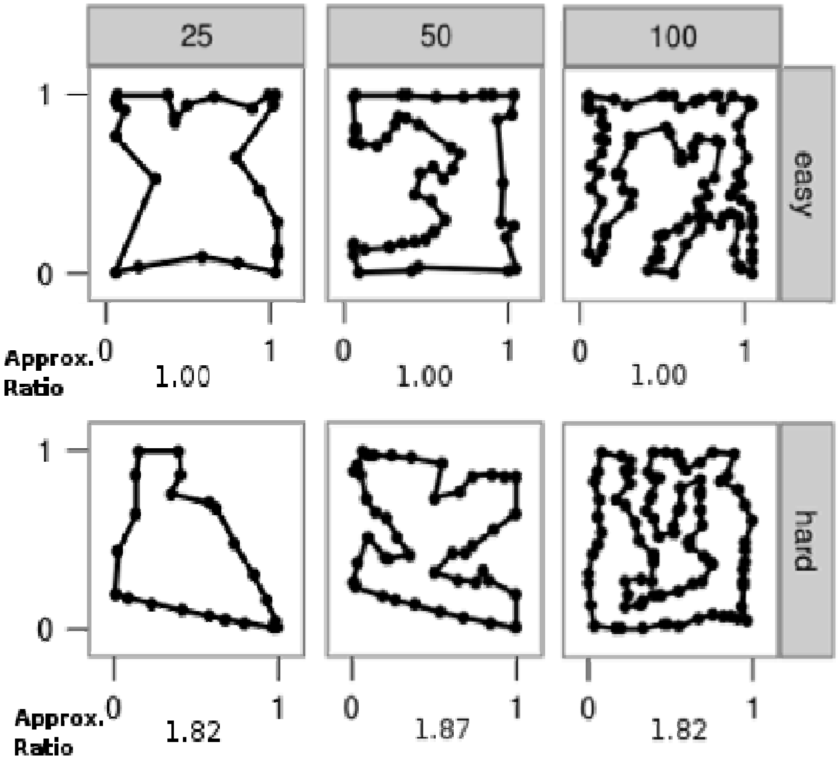


3/2-Approximation



Step 2: Analysis

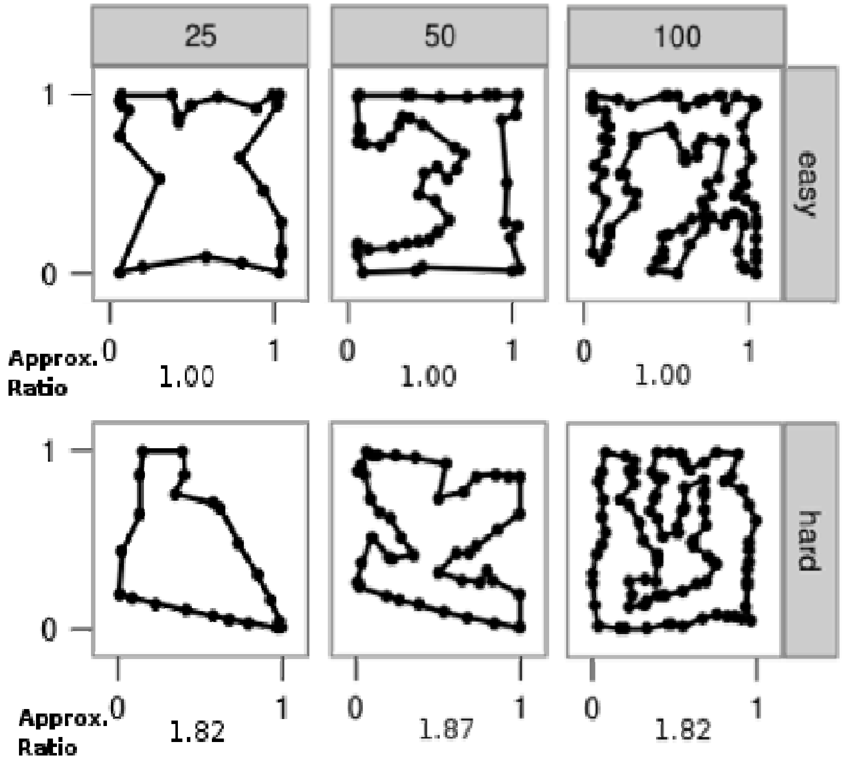
2-Approximation



- Distances of cities (optimal tour) are more uniform in the hard instances
- Standard deviations of the distances (optimal tour) of the easy instances are roughly twice as high than for the hard instances when considering small instance sizes. (decreases with *n*)
- Easy instances: small clusters (more uniform distribution in the hard instances)

Step 2: Analysis

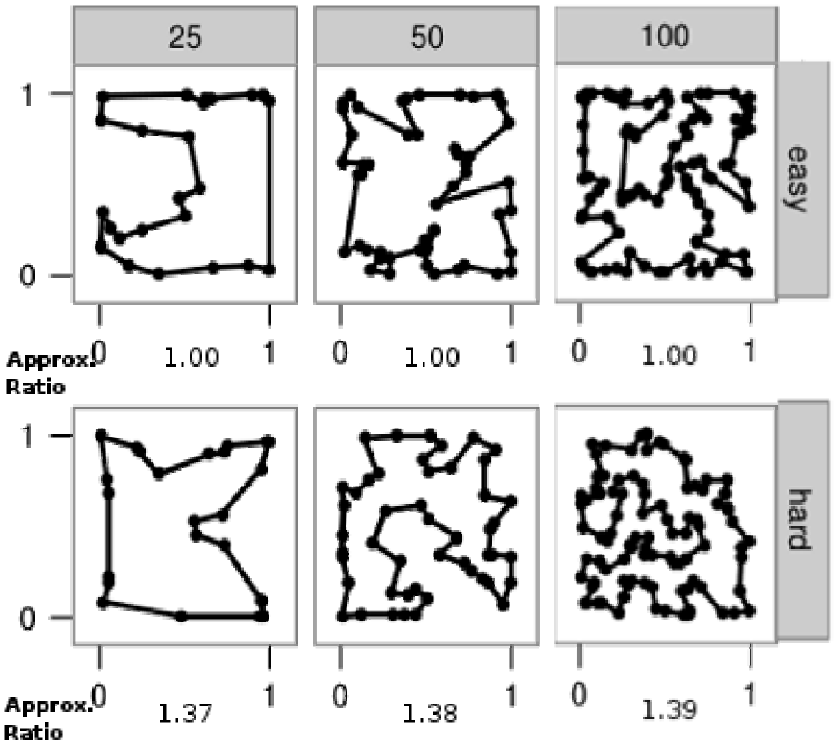
2-Approximation



- Easy instances: smaller angles (optimal tours)
- Mean angle values for easy/hard instances slightly decrease with the instance size.
- Instance shapes for small instances structurally differ from the respective shapes of larger instances. Consequently, the area covered by the convex hull is higher for larger instances.

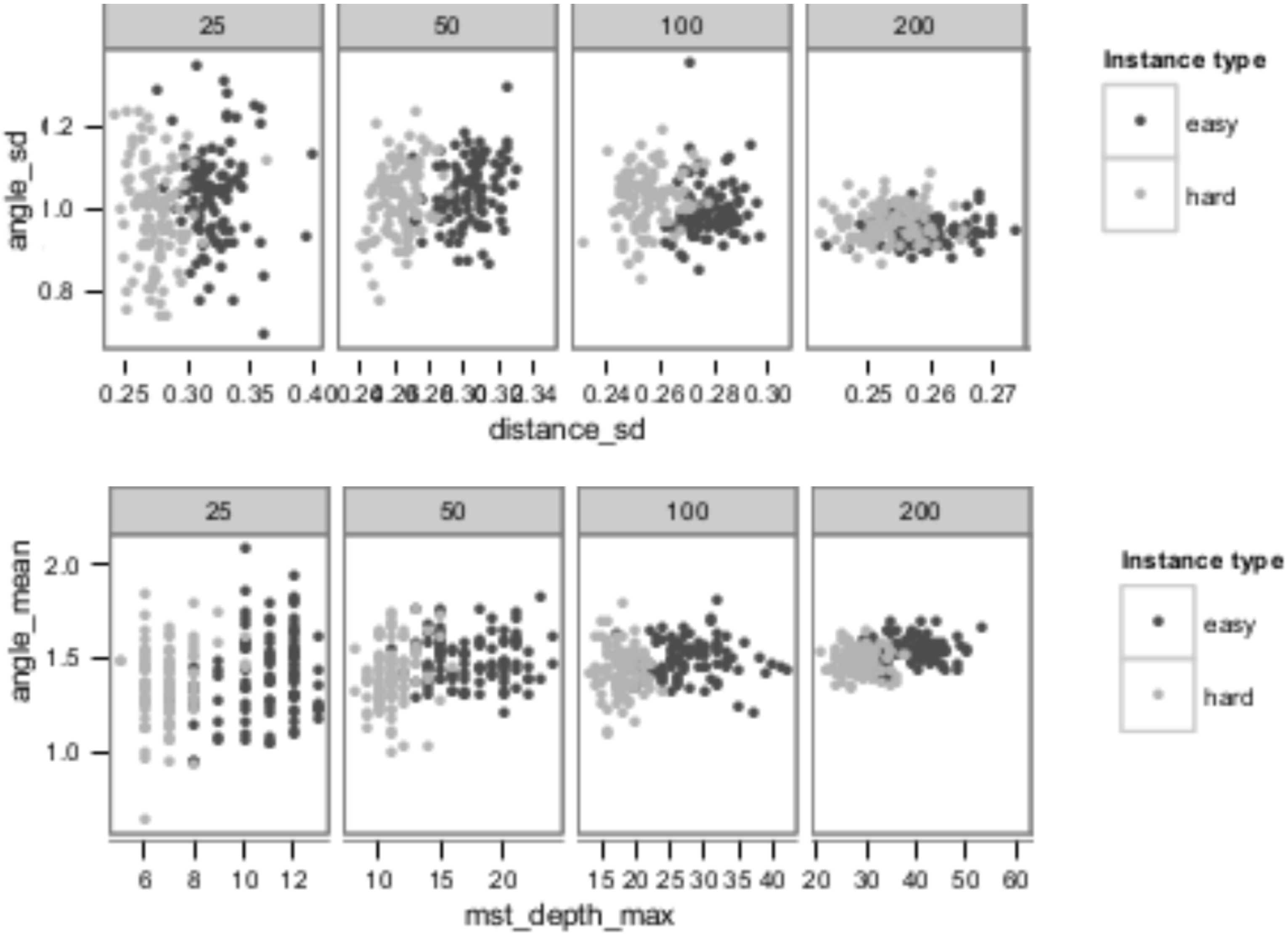
Step 2: Analysis

3/2-Approximation



- Visually, easy and hard instances do not differ significantly.
- Easy instances: considerably higher standard deviations of the distances (optimal tour).
- Mean angles: higher for (smaller) easy instances, lower for (larger) easy instances.

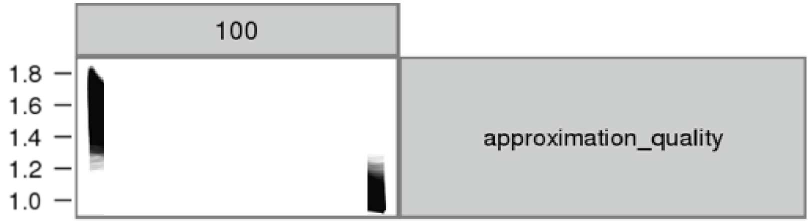
Exemplarily: Differentiating the generated Christofides Instances



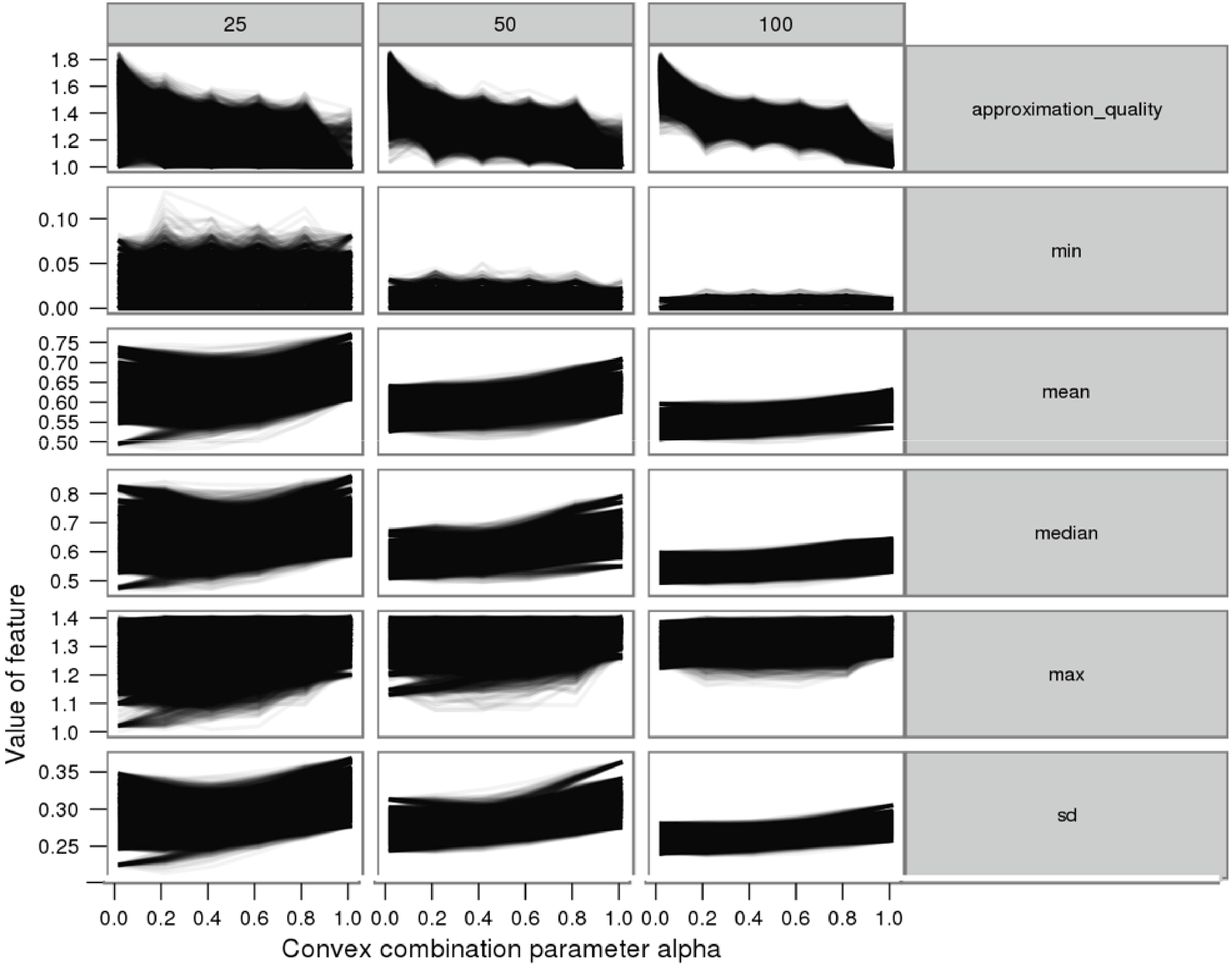
THE UNIVERSITY OF ADELAIDE AUSTRALIA

Morphed (Intermediate) Instances

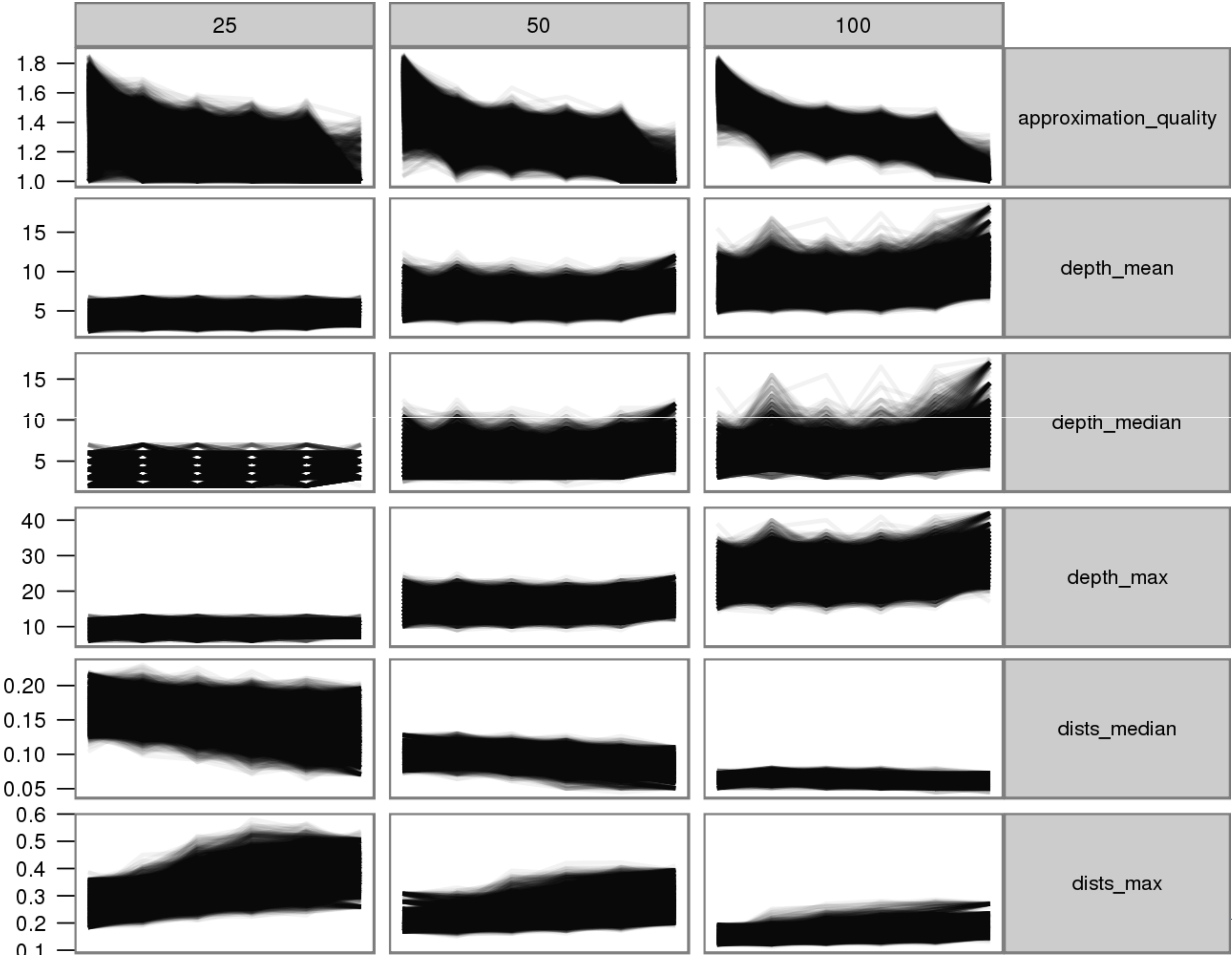
2-Approximation: distance features



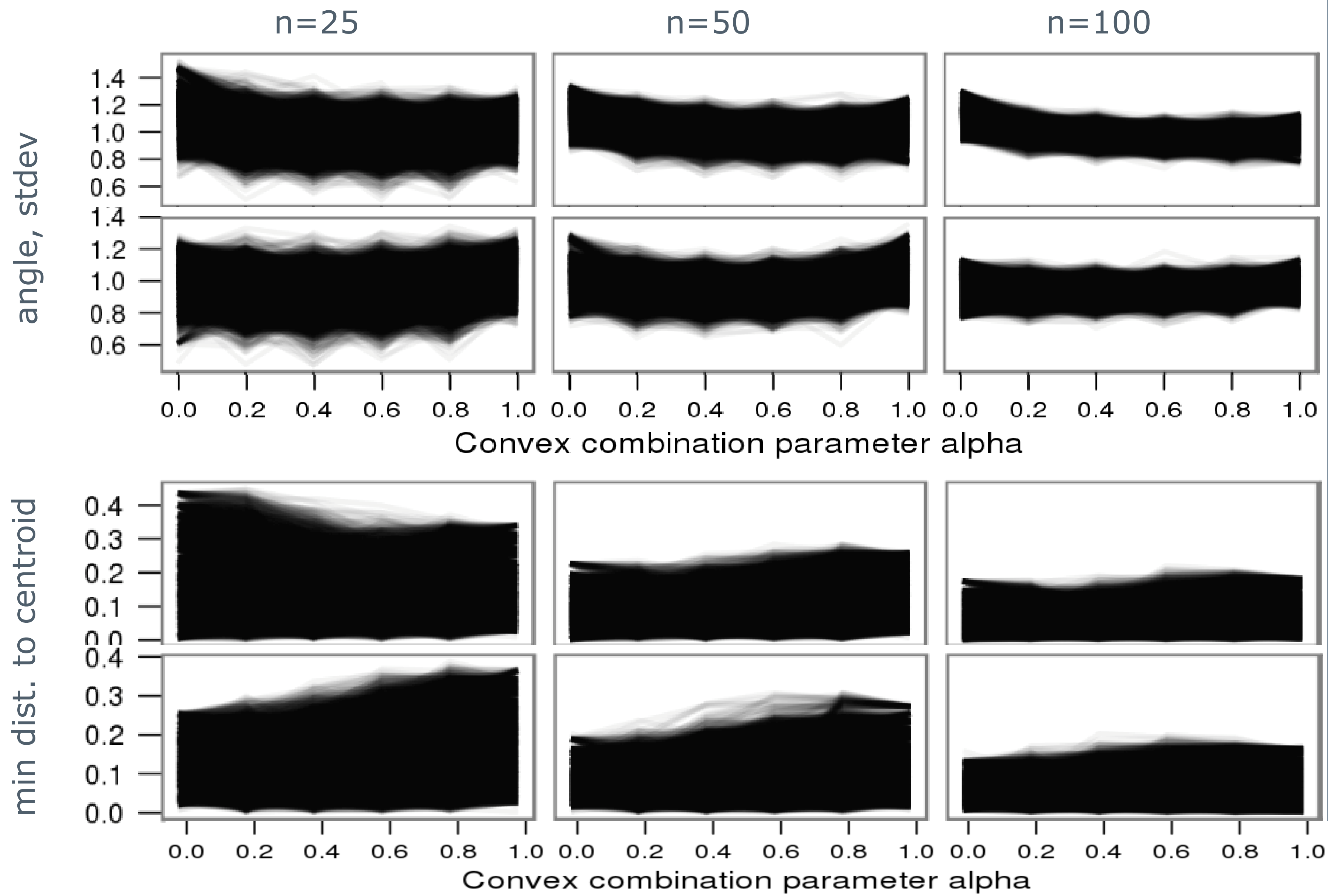
2-Approximation: distance features



2-Approximation: MST features



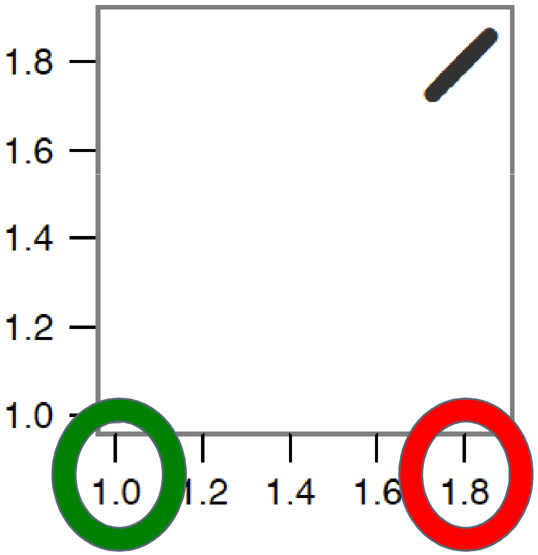
2-Approximation vs. 3/2-Approximation



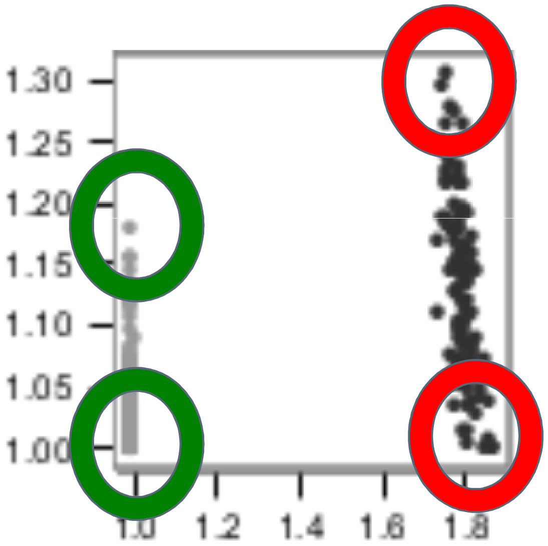
Cross-Comparison: 2-Approximation, 3/2-Approximation, 2-Opt

2-Approximation instances (n=25) on…

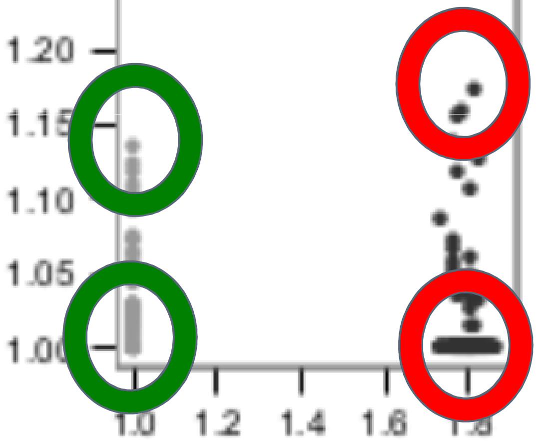
2-Approximation



3/2-Approximation

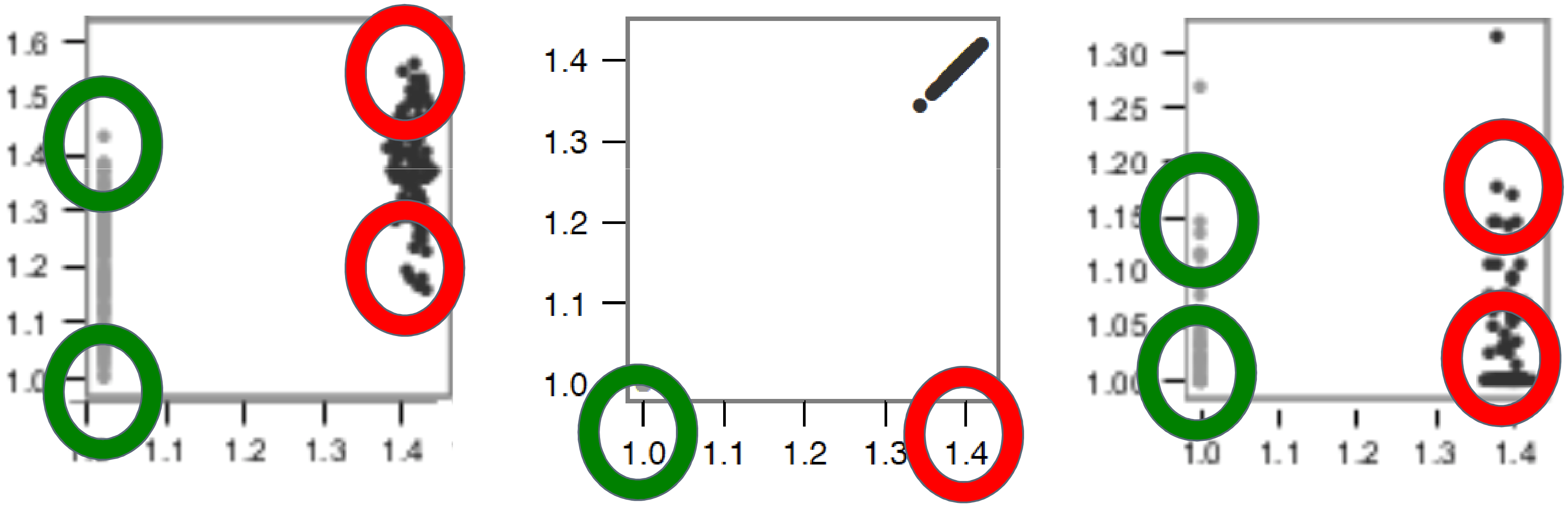


2-Opt



3/2-Approximation instances (n=25) on…

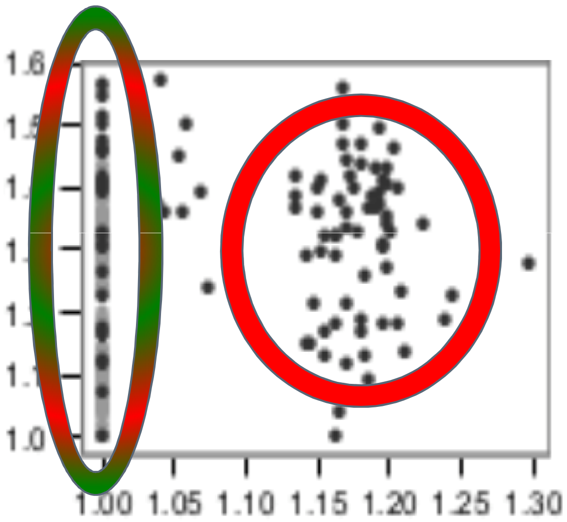
2-Approximation 3/2-Approximation 2-Opt



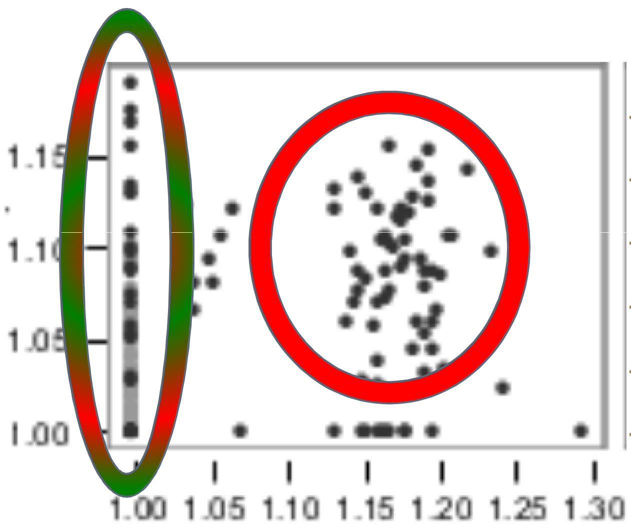
Note: 2-Opt “beats” 3/2-Approximation here (hard instances)

2-Opt instances (n=25) on...

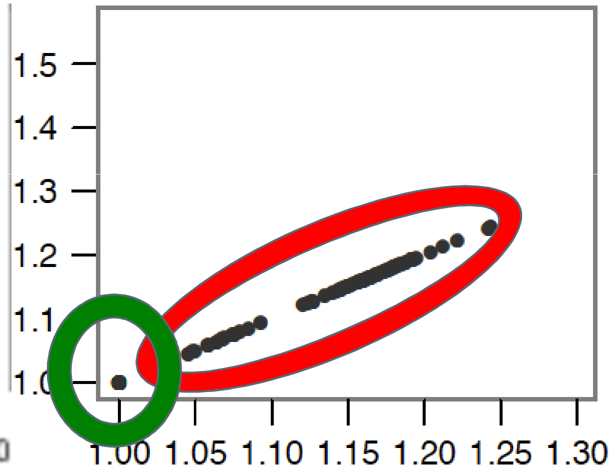
2-Approximation



3/2-Approximation



2-Opt



Note: 3/2-Approximation “beats” 2-Opt here (hard instances)

Thank you!