Local Search and the Traveling Salesman Problem: A Feature-Based Characterization of Problem Hardness

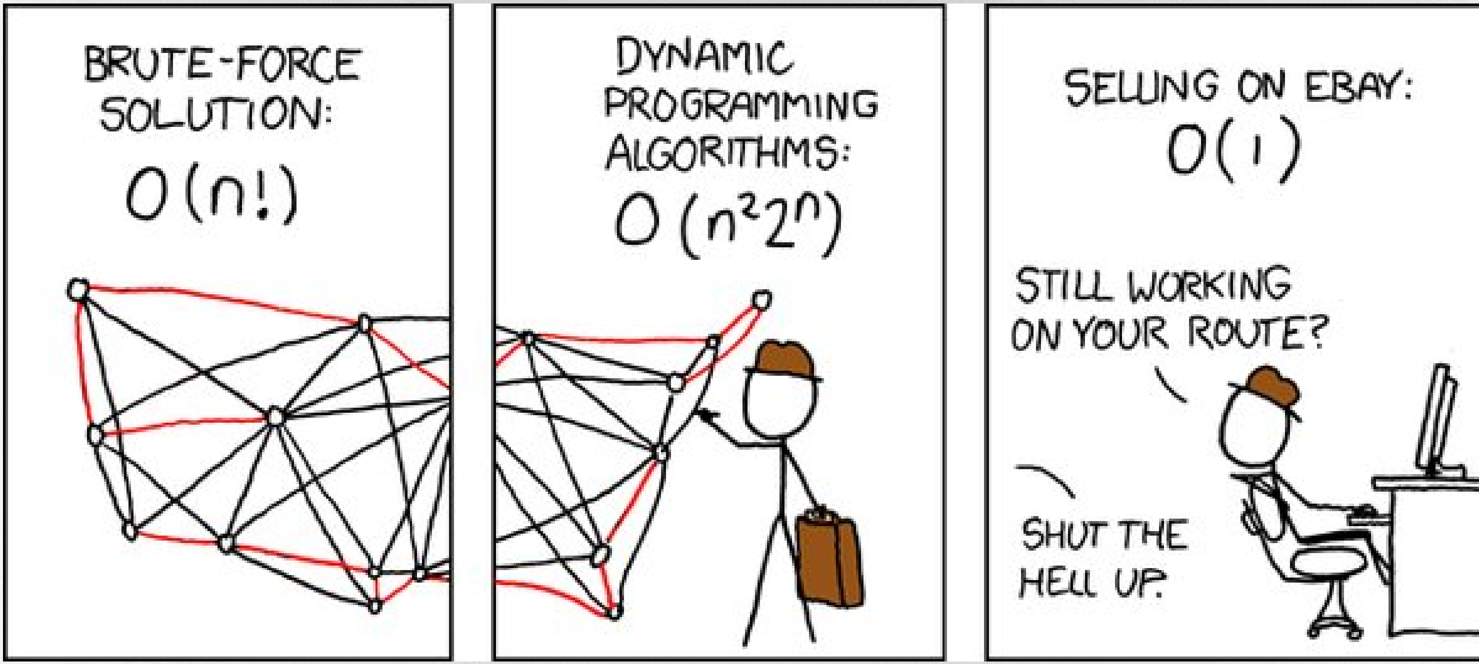
Olaf Mersmann, Bernd Bischl, Jakob Bossek and Heike Trautmann

Department of Statistics, TU Dortmund University, Germany

Markus Wagner and Frank Neumann

School of Computer Science, The University of Adelaide, Australia

The Traveling Salesman Problem (TSP)



Aim: Predict Hardness of TSP instances

Problem Hardness: Two options

Number of swaps/iterations/...

Used in Smith-Miles et al. (2010)

Approximation quality

$$= \frac{\text{Expected solution tour length}}{\text{Optimal tour length}}$$

Characterize TSP instances

Requirement

All features can be computed without knowledge of the optimal tour.
Eliminates some (interesting) features.

Challenges

Normalization, dependence on # of nodes / edges

Characterize TSP instances

Taken from literature

Literature used

Smith-Miles et al. (2010), Kanda et al. (2011) and Smith-Miles and van Hemert (2011)

Classes of features

- Nearest Neighbor Distance (NNDs)
- Clustering
- Edge Costs / Distance Matrix

Focus on 2-opt (Croes, 1958) algorithm.

Reasons

- ▷ Historically first successful local search method for TSP
- ▷ Easy to understand
- ▷ Some progress on theoretical analysis
  (Chandra et al., 1999 and Englert et al., 2007)

Where do the TSP instances come from?

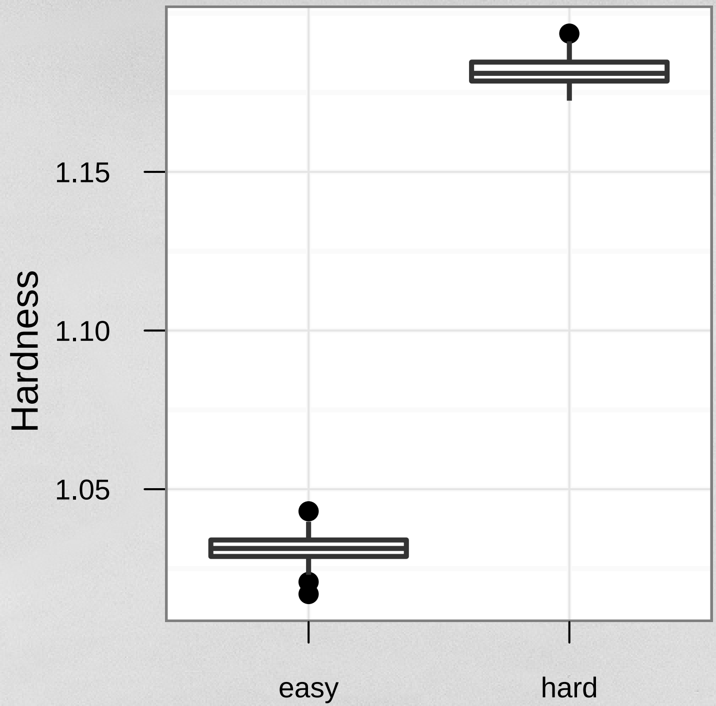
Instance Generator: EA

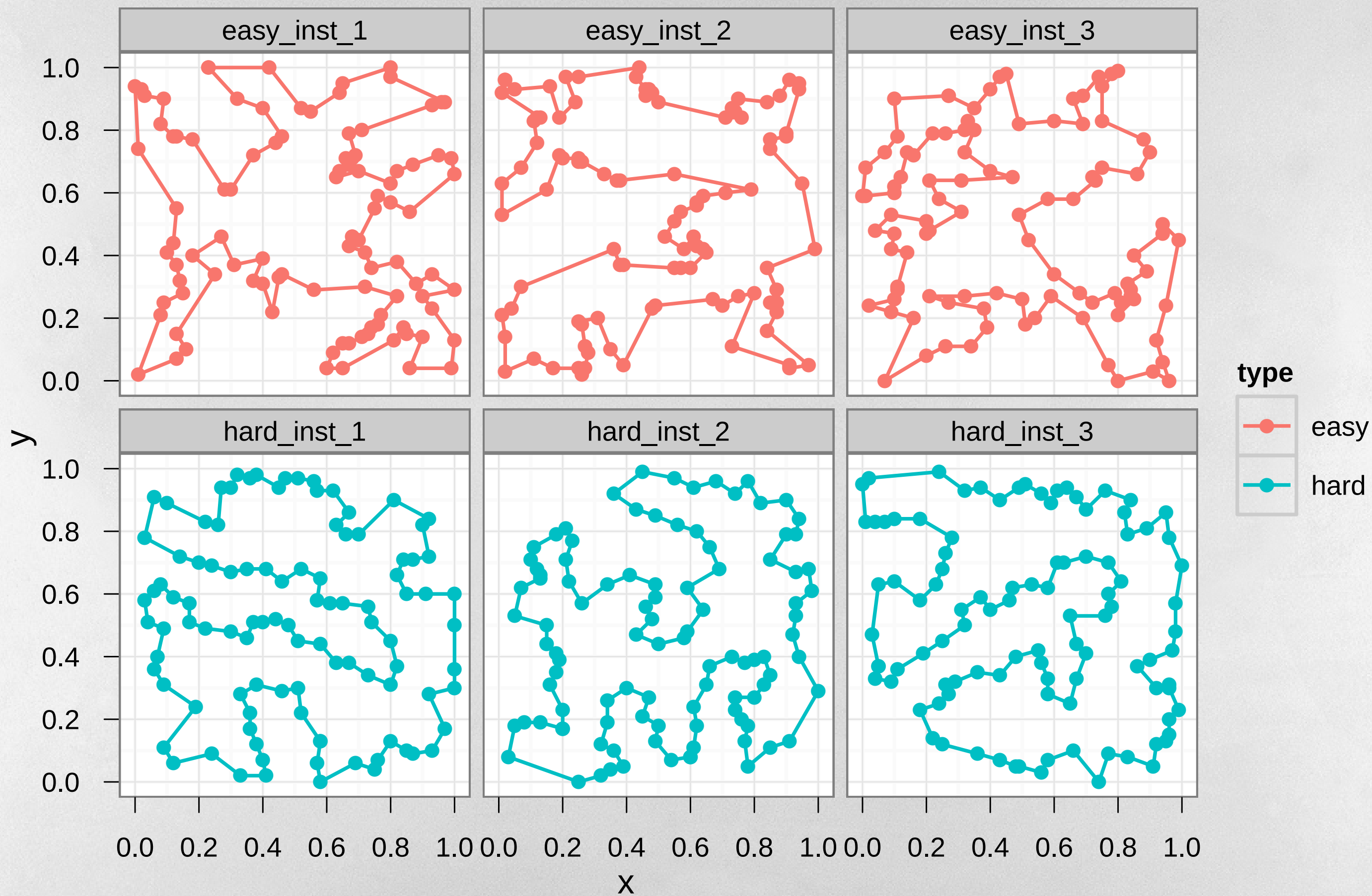
```
function tsp_generator(popSize=30, instSize=100, poolSize=50,
                       digits=2, repetitions=500):
  pop = randomInstances(popSize, instSize)
  while not done:
    fitness = computeFitness(pop, repetitions)
    matingPool = tournamentSelection(pop, poolSize, fitness)
    nextPop[1] = pop[whichBest(fitness)]
    for k = 2 to popSize:
      parent1, parent2 = randomElements(2, matingPool)
      offspring = uniformCrossover(parent1, parent2)
      nextPop[k] = round(
        uniformMutation(normalMutation(offspring)), digits)
    pop = nextPop
```

Use EA to generate 100 easy and hard instances

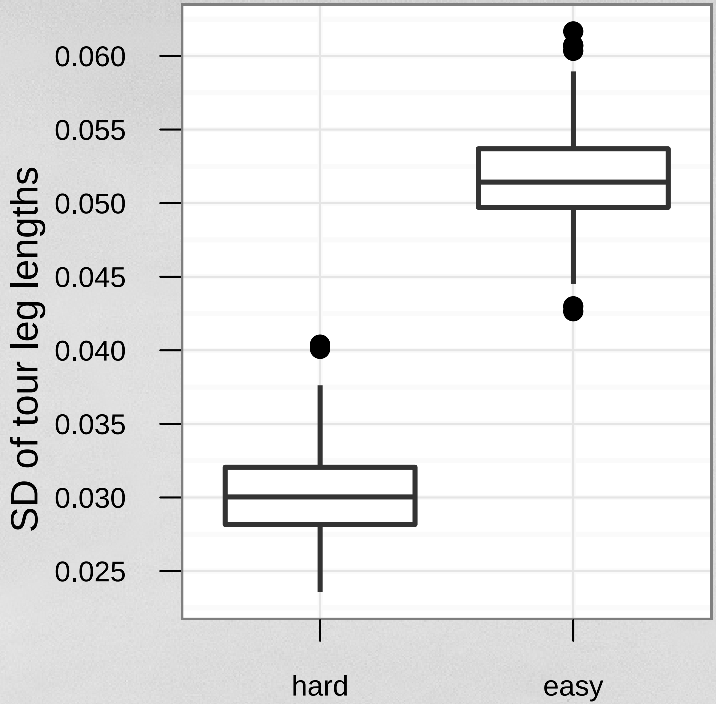
Problems

- ▷ Fitness function expensive
- ▷ Lots of manual tuning of EA
- ▷ Some runs hung


Hardness
1.15
1.10
1.05
easy
hard


easy_inst_1
easy_inst_2
easy_inst_3
hard_inst_1
hard_inst_2
hard_inst_3
1.0
0.8
0.6
0.4
0.2
0.0
y
x
type
easy
hard

Observation



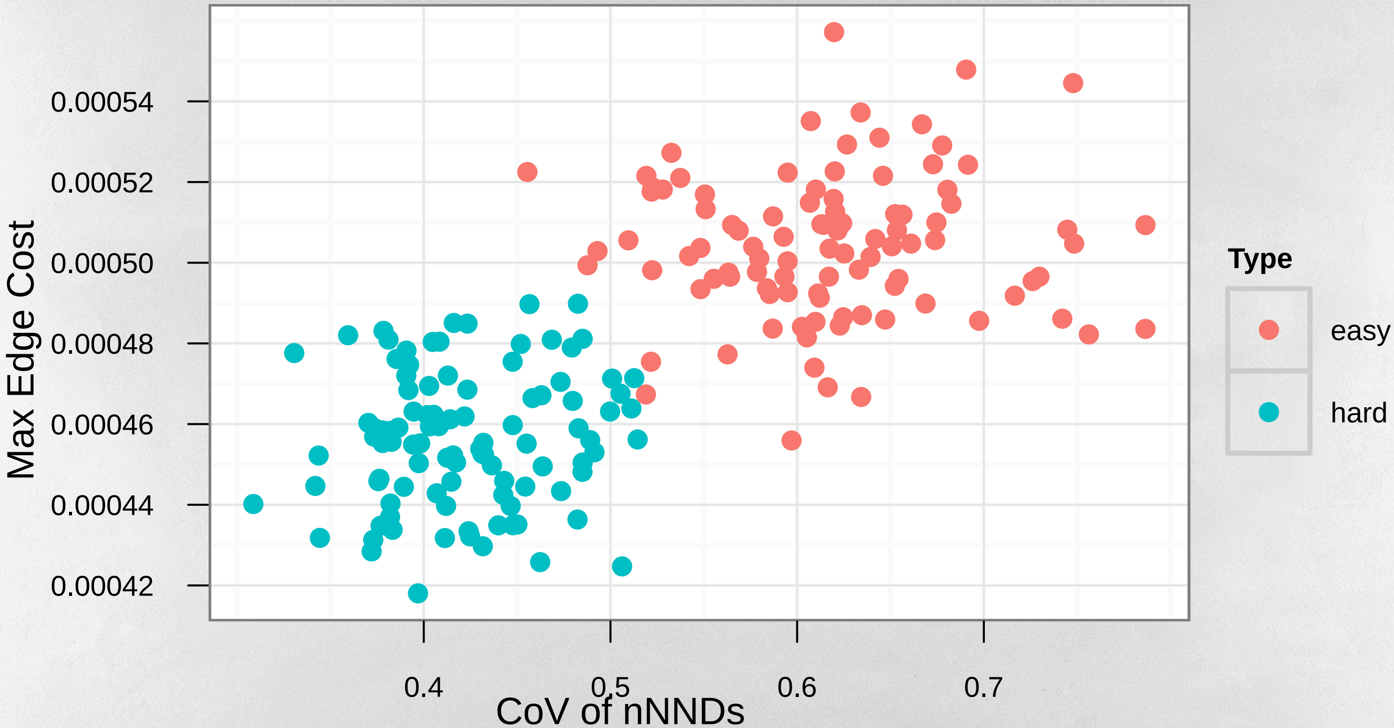
1 Tour leg lengths differ less for hard instances.

Prediction

- ▷ Calculate all features for the 200 instances
- ▷ Use decision tree (CART) to predict instance type

```
coefficient_of_variation_of_nnds >= 0.5167739 → easy
coefficient_of_variation_of_nnds < 0.5167739
   highest_edge_cost >= 0.000485 → easy
   highest_edge_cost < 0.000485 → hard
```

10-fold CV error rate: 3.02%



Max Edge Cost
0.00054
0.00052
0.00050
0.00048
0.00046
0.00044
0.00042
0.4
0.5
0.6
0.7
CoV of nNNDs
Type
easy
hard

This was an ``easy” task.

Instances chosen to be maximally different!

Morphing instances

We are missing instances that are between the two classes.

Idea

Create convex combination of an easy I_e and a hard instance I_h

$$I_n = \alpha I_e + (1 - \alpha) I_h \quad \text{with} \quad \alpha \in [0,1]$$

Morphing instances

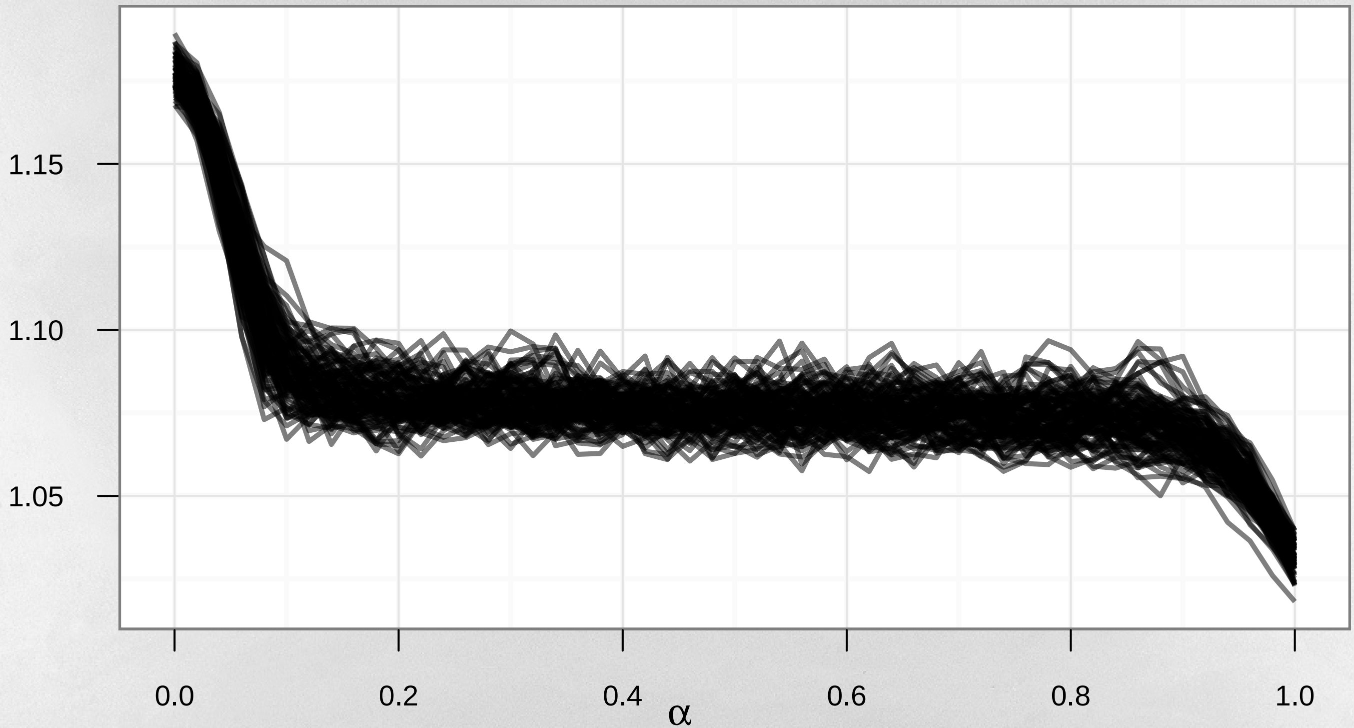
Possible Improvements

Match up points to minimize movement

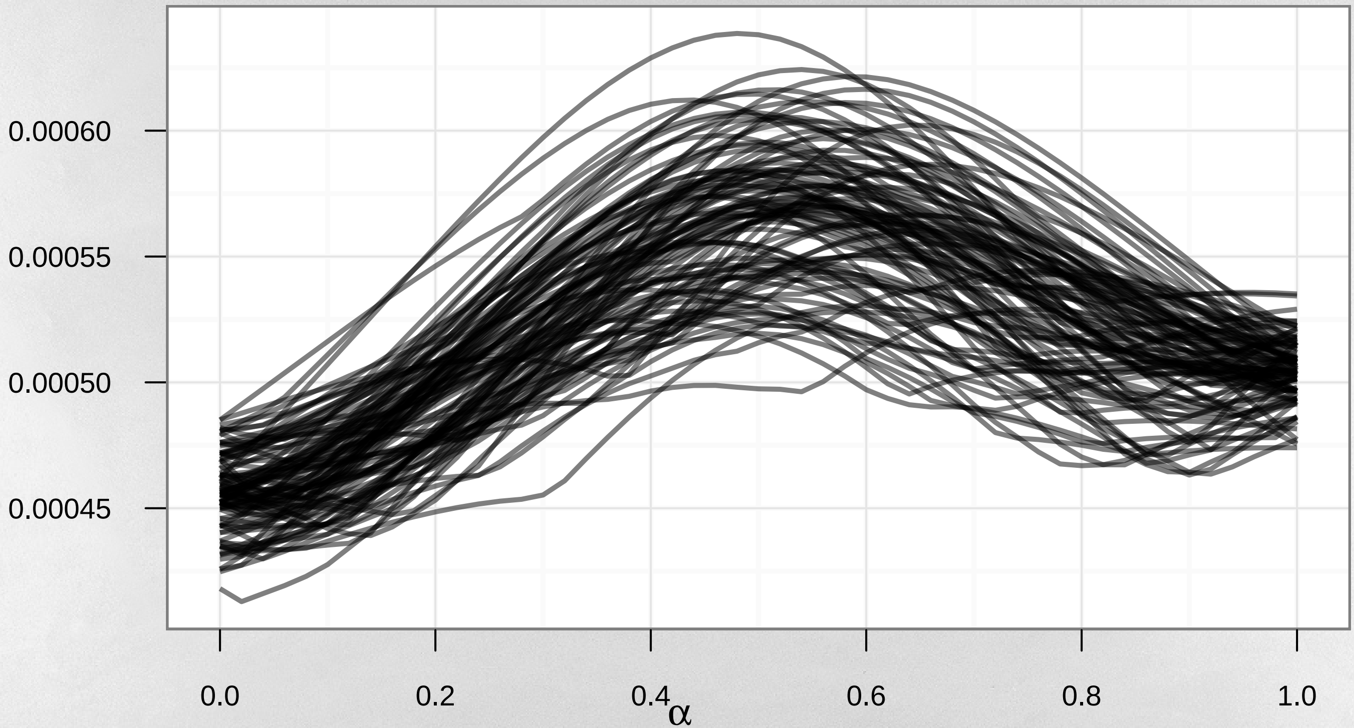
Usage

- ▷ For every combination of instances generate morph
- ▷ Calculate features for different α $(0.2, 0.4, ..., 0.8)$

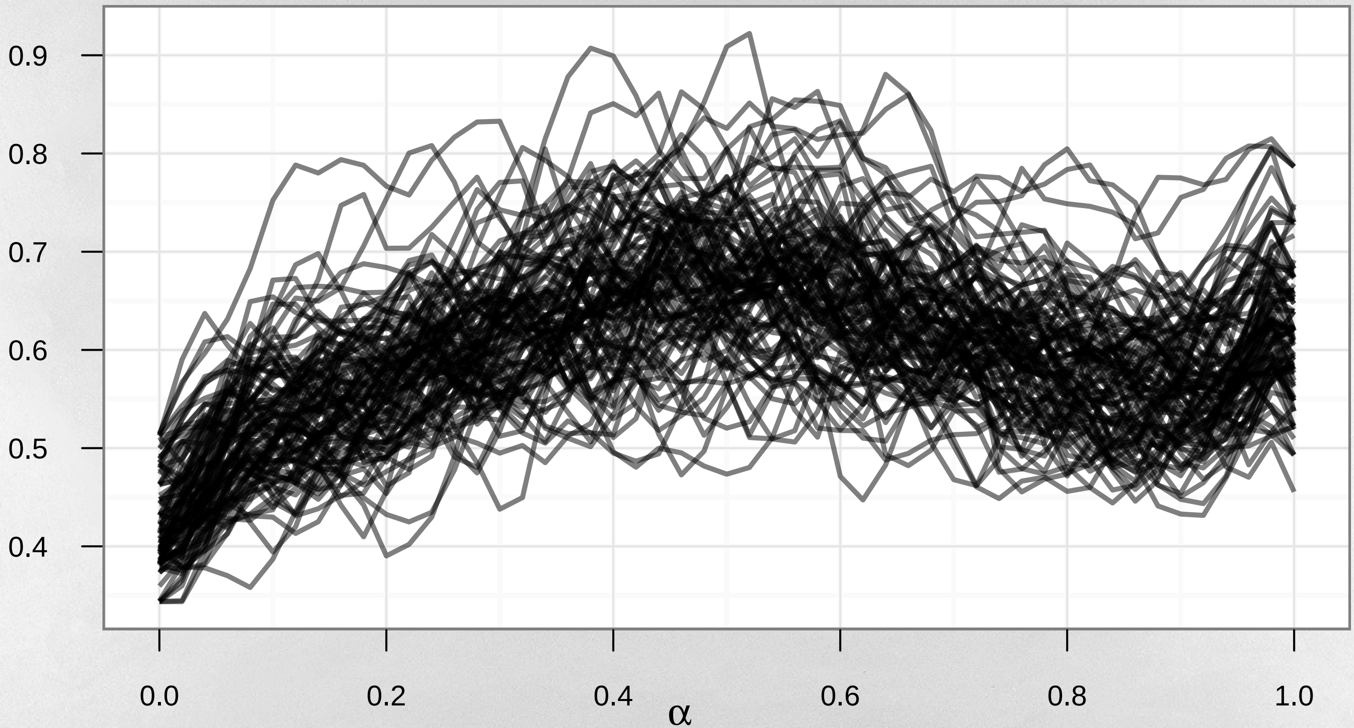
Problem Hardness



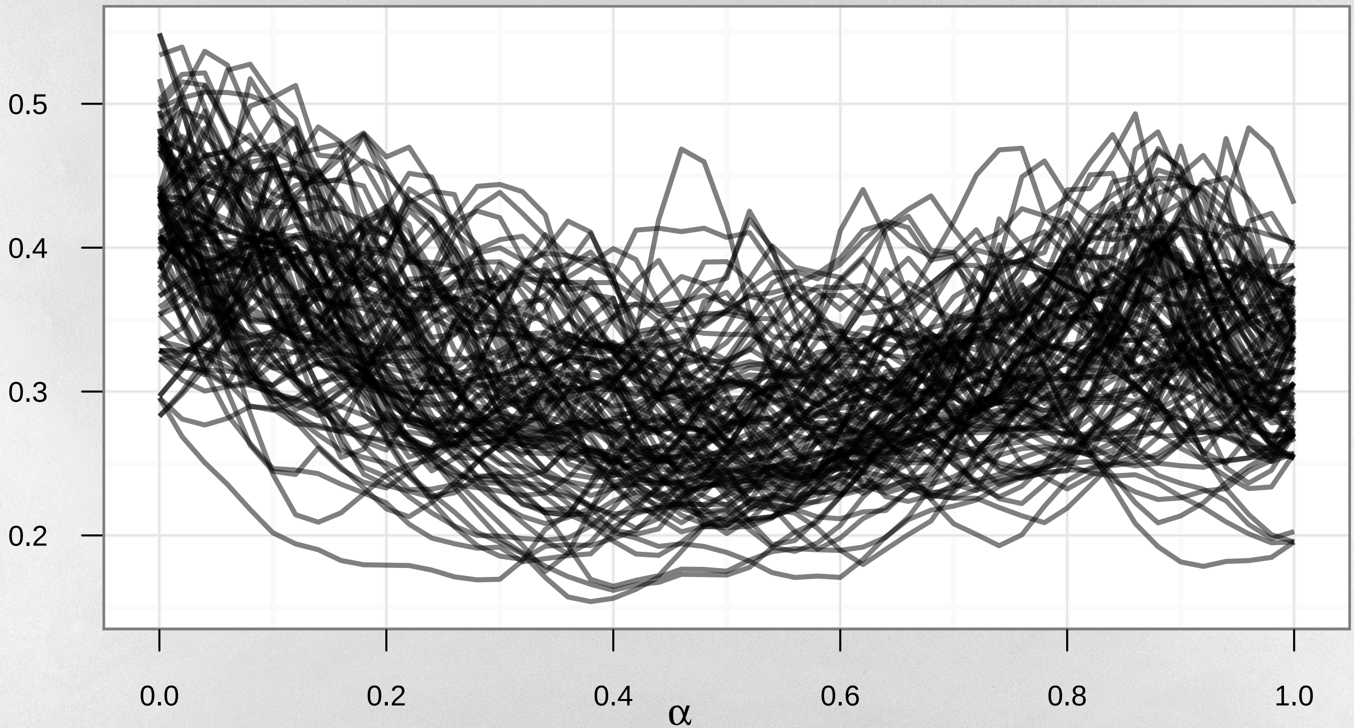
Max Edge Cost



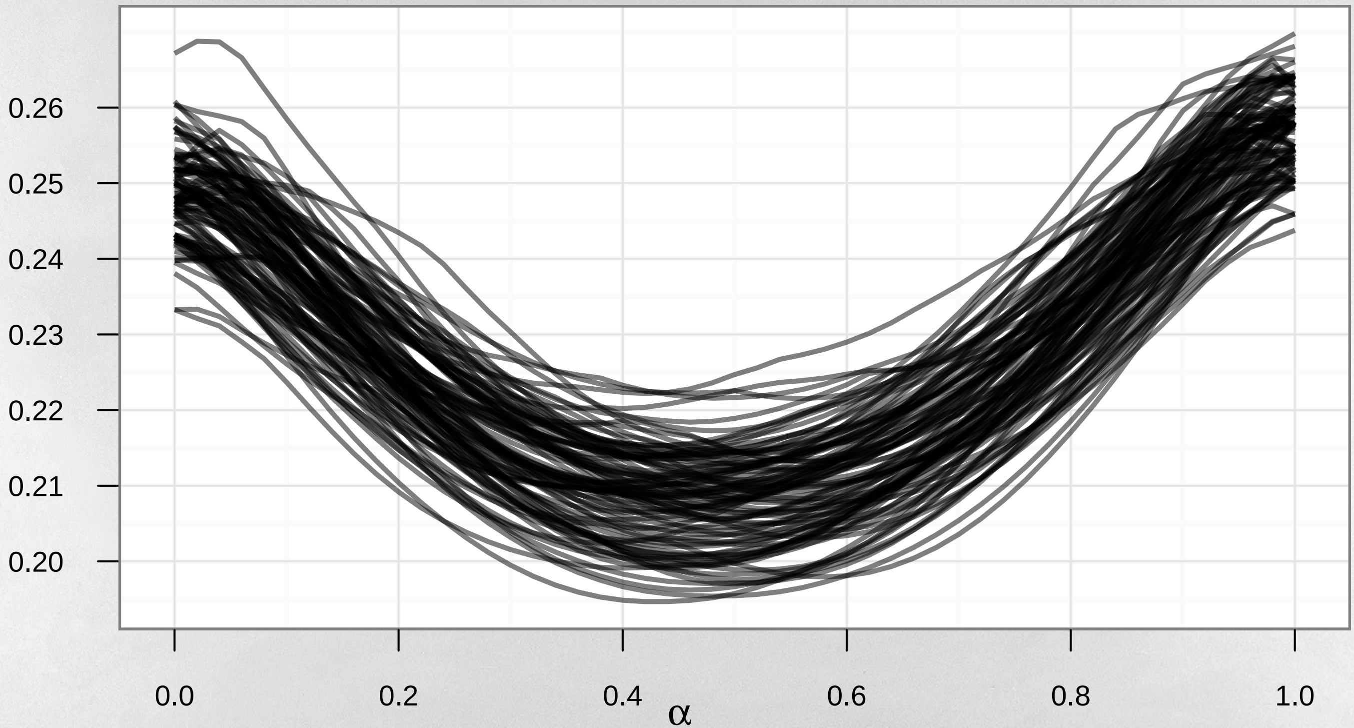
CoV of nNNDs



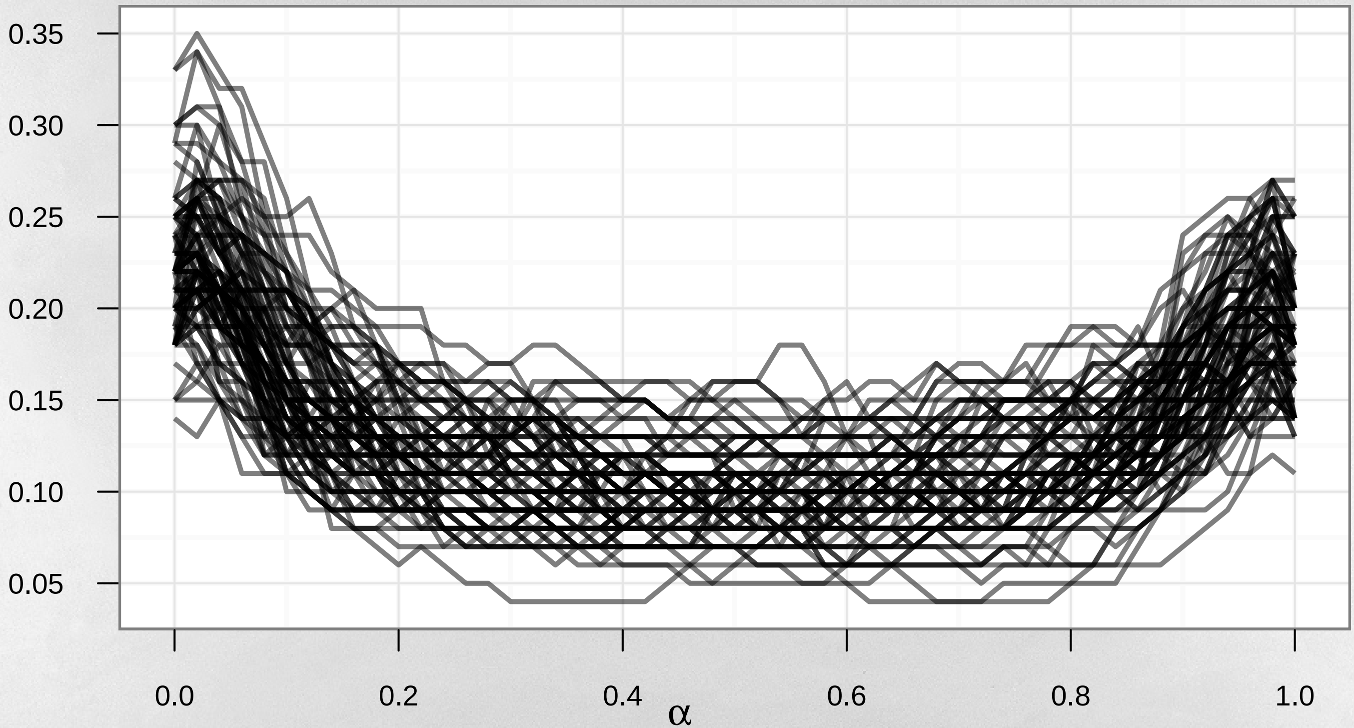
Mean of nNNDs



Variation of Edge Cost



Ratio of Cities near Edge



Prediction

Fit MARS model to data.

- ▷ Only use subset of morph results
- ▷ Do SFS to select subset of variables

RMSE estimated via 3-fold CV: 0.0113

Interpretation

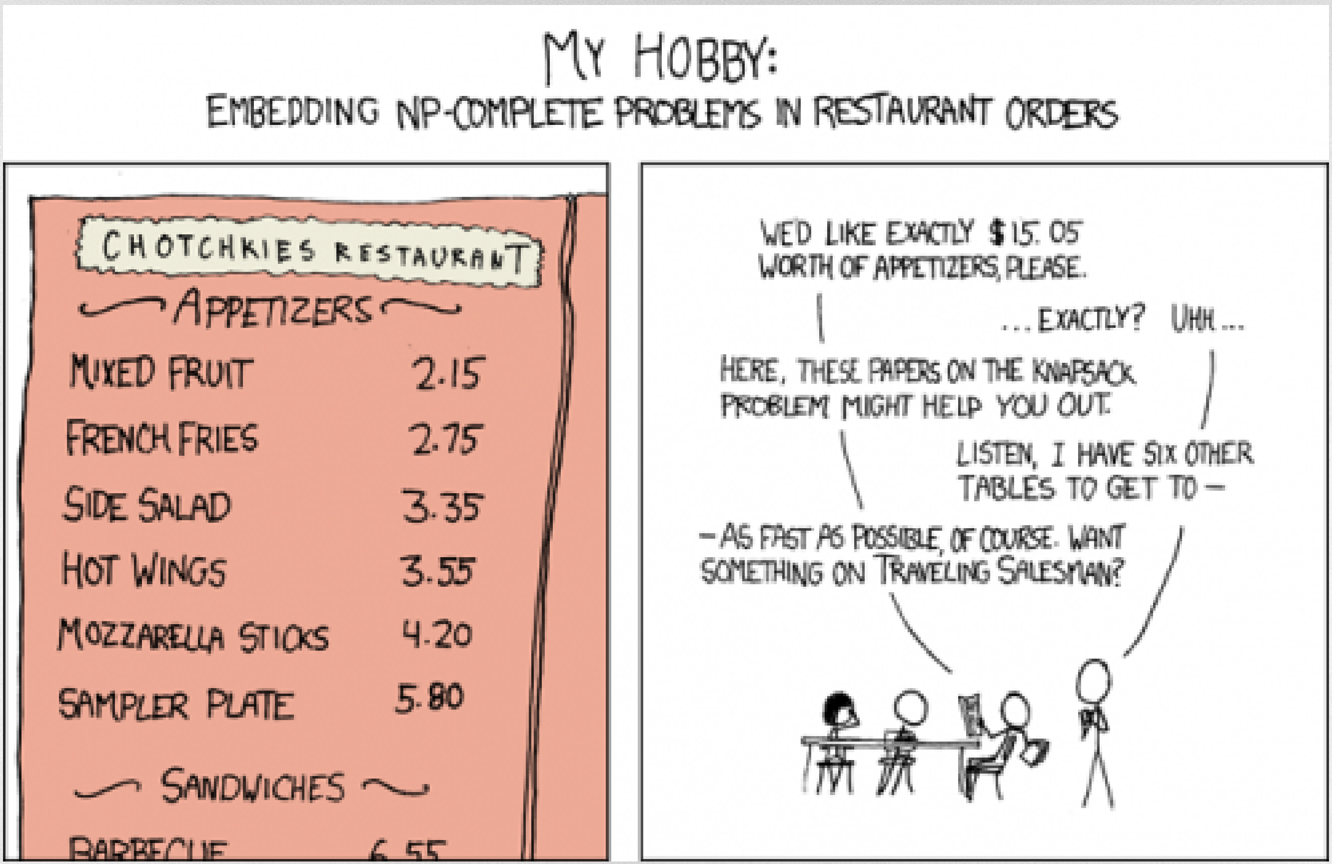
Not a black-box model. Please see paper for plots and interpretation.

Conclusion

- ▷ Generated ``easy” and ``hard” instances for 2-opt heuristic
- ▷ Characterized the instance sets using easily calculated features
- ▷ Showed novel approach to generate ``medium” instances (morphing)
- ▷ Predicted hardness of instance based on features using simple models

Outlook

- ▷ Optimize instance generation
- ▷ Study relation between features and theoretical properties of 2-opt
- ▷ Improve morphing
- ▷ Generate more diverse instance sets


MY HOBBY:
EMBEDDING NP-COMPLETE PROBLEMS IN RESTAURANT ORDERS
CHOTCHKIES RESTAURANT
APPETIZERS
MIXED FRUIT 2.15
FRENCH FRIES 2.75
SIDE SALAD 3.35
HOT WINGS 3.55
MOZZARELLA STICKS 4.20
SAMPLER PLATE 5.80
SANDWICHES
WED LIKE EXACTLY $15.05 WORTH OF APPETIZERS, PLEASE.
...EXACTLY? UHH...
HERE, THESE PAPERS ON THE KNAPSACK PROBLEM MIGHT HELP YOU OUT.
LISTEN, I HAVE SIX OTHER TABLES TO GET TO –
– AS FAST AS POSSIBLE, OF COURSE. WANT SOMETHING ON TRAVELING SALESMAN?