http://cs.adelaide.edu.au/~markus/
The slides will be made available today.

Markus Wagner
markus.wagner@adelaide.edu.au

Approximation-Guided Many-Objective Optimization and the Travelling Thief Problem

Anhui University and
IEEE CIS Chapter Hefei

Optimisation and Logistics

Algorithmic Game Theory

Coordinator:
Dr Mingyu Guo

Renewable Energy

Coordinator:
Dr Markus Wagner

Foundations of Heuristics

Coordinator:
Prof Frank Neumann

Staff Profile:

6 faculty members
2 postdocs
8 PhD students

Search-based Software Engineering

Coordinator:
Dr Bradley Alexander

Supply Chain Management

Coordinator:
Dr Sergey Polyakovskiy

Optimisation and Logistics

Supply Chain Management (Australian Research Council funded)

- Large scale industrial optimisation problems with many interacting components.

Dynamic Constraints (ARC funded)

- Algorithms for problems with dynamically changing constraints.

Dynamic Adaptive Software Configurations (ARC funded)*

- Self-adapt system configurations to changing conditions.

Lots of other knowledge, either in-house or via international collaborations, e.g. more theory, system modelling, speed-up of simulations (algorithmically or using machine learning)…

Some of the activities of Optimisation and Logistics 2016-2018

- ACM Genetic and Evolutionary Computation Conference 2016
  (General Chair: Frank Neumann)
- NII Shonan Meeting on "Computational Intelligence for Software Engineering, Shonan Village Centre, Japan.
  Organizers: Hong Mei (Peking), Frank Neumann (UoA), Xin Yao (Birmingham)
- Dagstuhl Seminar on "Automatic Algorithm Selection and Configuration", Schloss Dagstuhl, Germany
  Organizers: Heike Trautmann (Muenster), Holger Hoos (Vancouver), Frank Neumann (UoA).
- NII Shonan Meeting on "Data-Driven Search-Based Software Engineering", Shonan Village Centre, Japan.
  Organizers: Markus Wagner (UoA), Leandro Minku (Leicester), Ahmed E. Hassan (Queens U), John Clark (York)
- Australasian Conference on Artificial Life and Computational Intelligence 2018
  (General Chair: Markus Wagner)
- International Workshop on Benchmarking of Computational Intelligence Algorithms, BOCIA, http://iao.hfuu.edu.cn/bocia18
  (Co-Chair: Markus Wagner)

Markus Wagner

2003-2009 Universität Koblenz · Landau

2006-2007 The University of Georgia

2010-2013 max planck institut informatik

2013 The University of Adelaide — Senior Lecturer

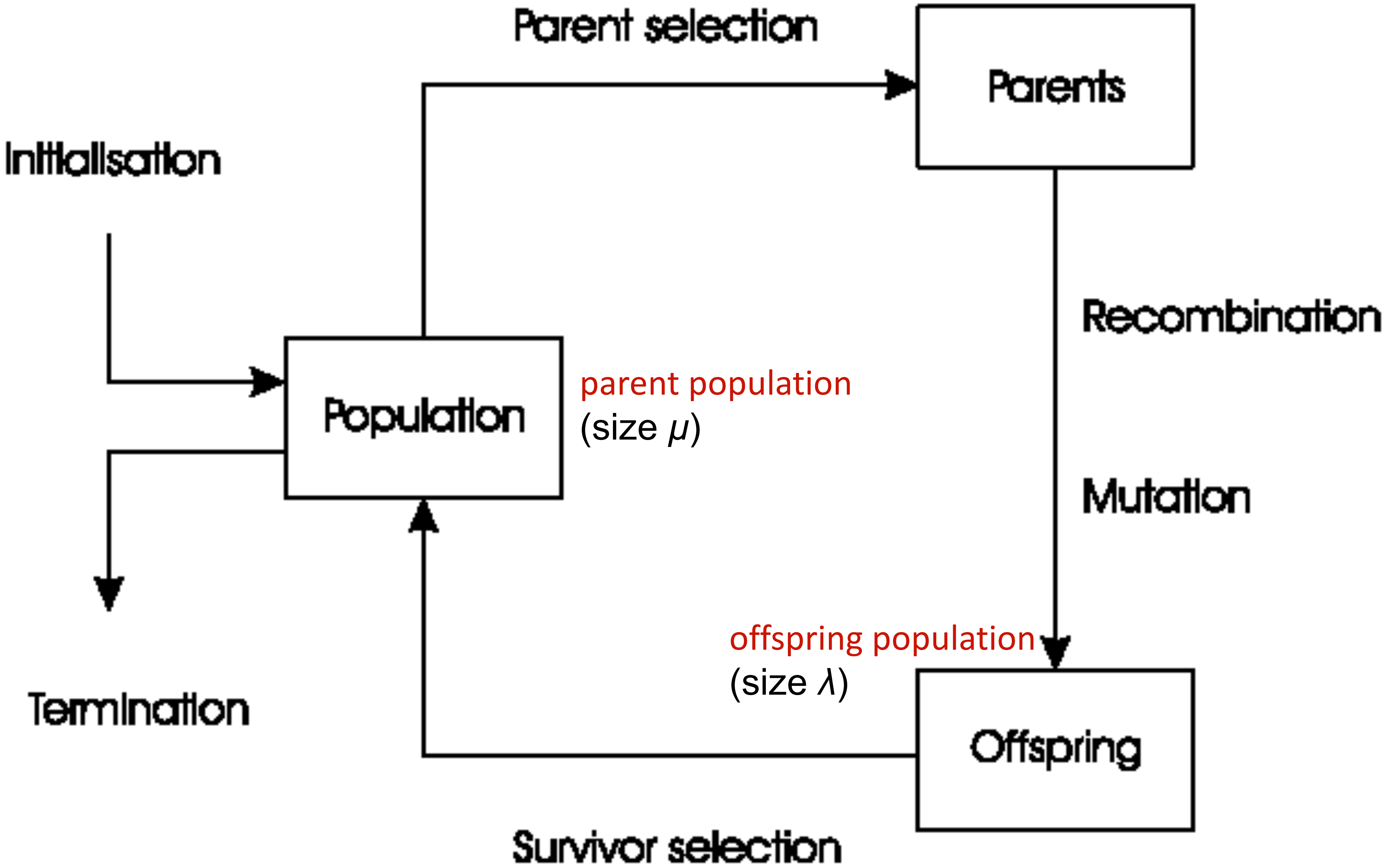
Summary: 80+ papers/co-authors/reviews/events/...
1 best paper/poster/presentation/keynote/medal/...
2nd time in Hefei ☺

IEEE CIS: Chair University Curricula 2016/2017
Chair Educational Material Subcommittee 2014/2015
Founding Chair of Task Force ”CI in the Energy Domain”

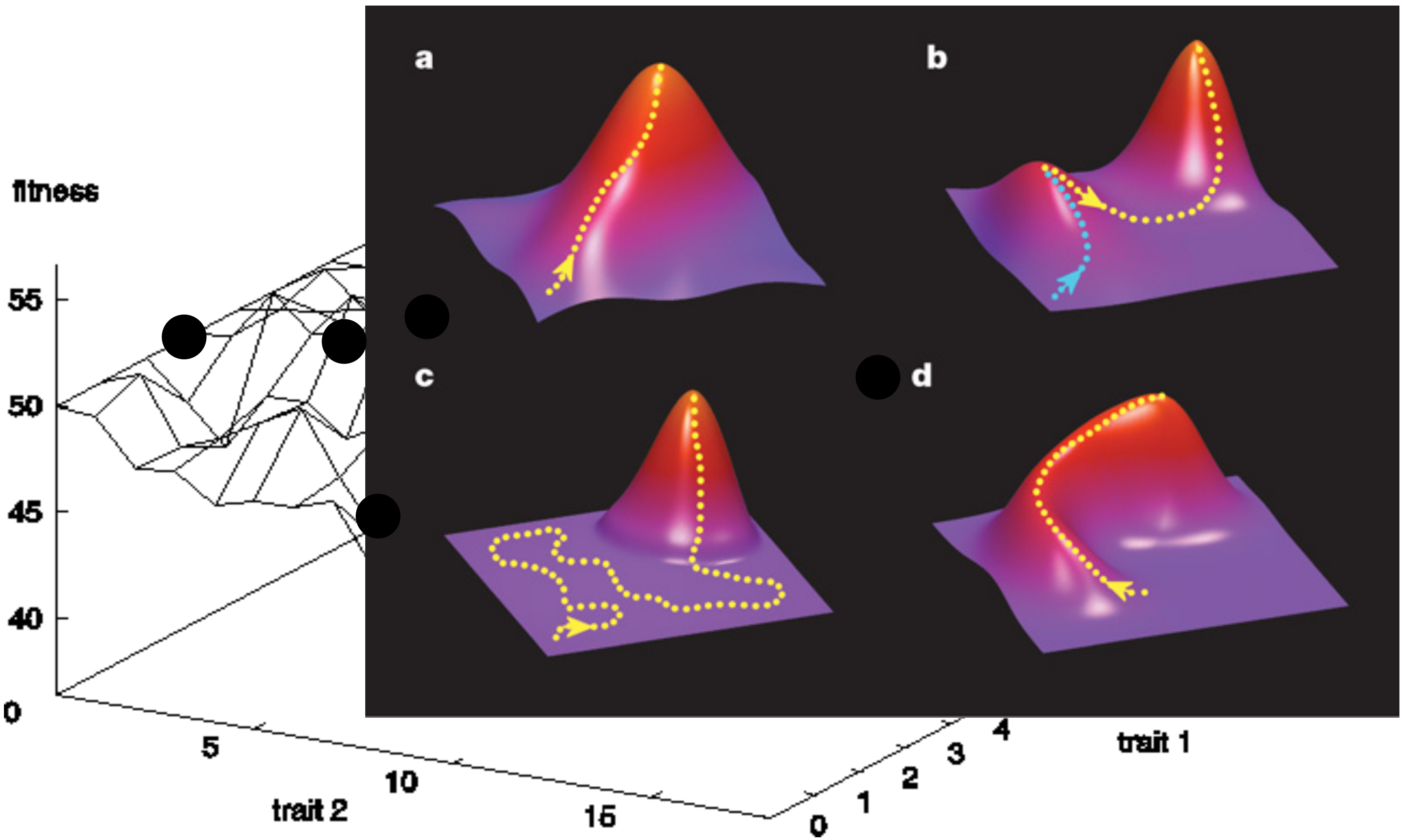
Approximation-Guided Evolutionary Multi-Objective Optimization

Joint work with Frank Neumann (U Adelaide), Karl Bringmann (ETH Zurich), Tobias Friedrich (Hasso Plattner Institute)

Evolutionary Algorithms: Darwin's "survival of the fittest"



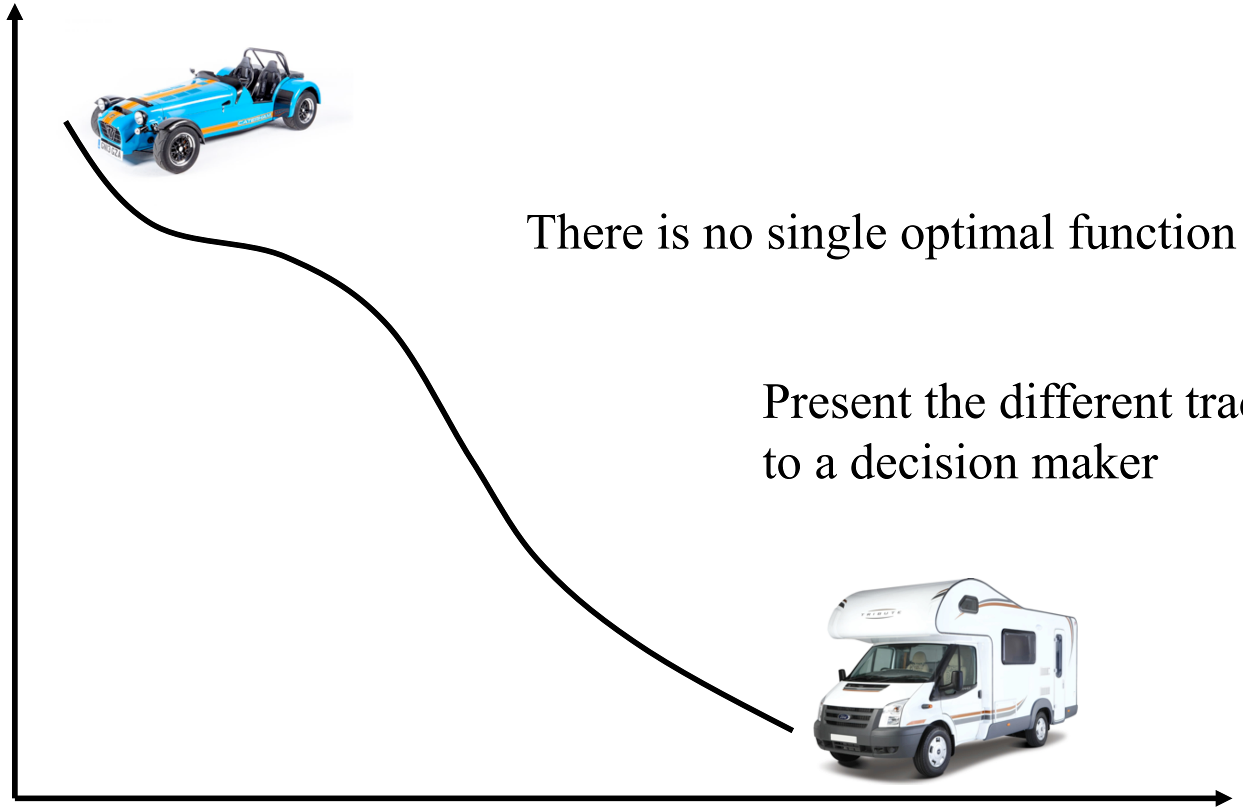
Example with two decision variables



Multi-Objective Optimisation

Many problems have more than one goal function

Example: Buying a new car

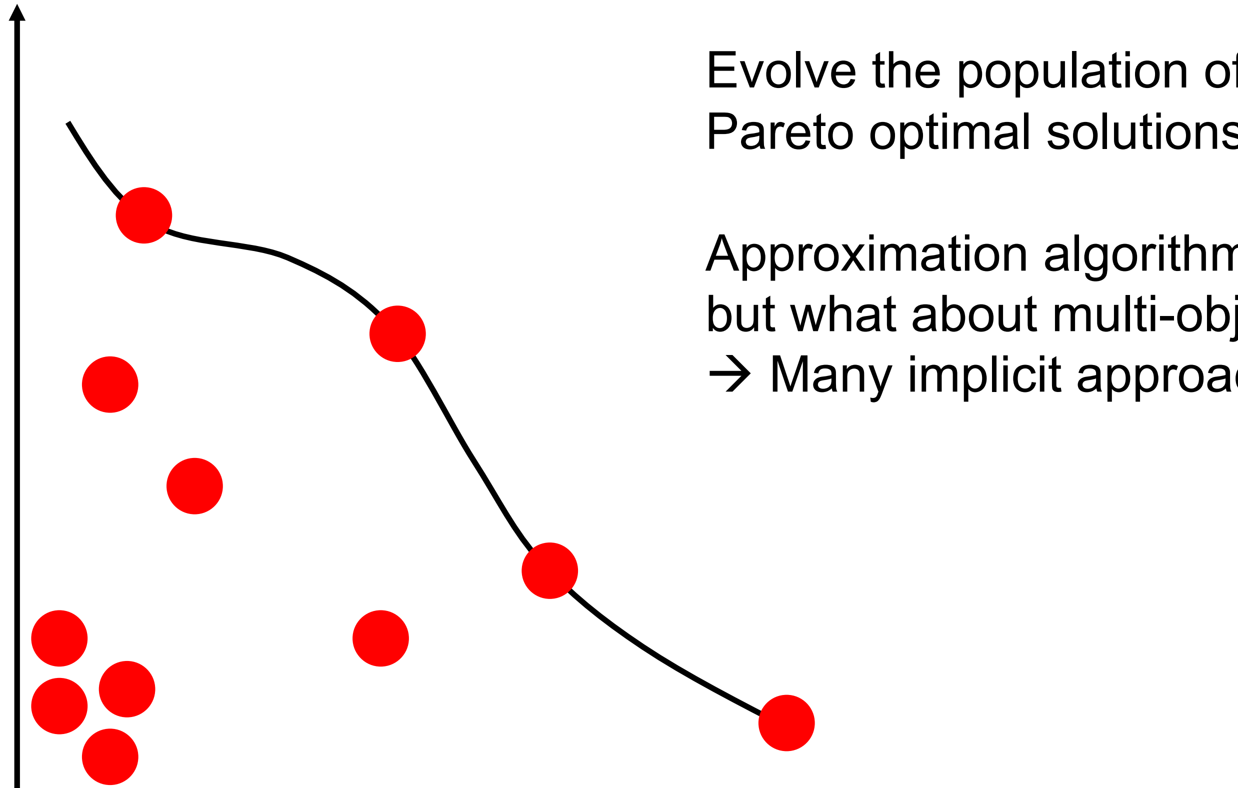


There is no single optimal function value

Present the different trade offs
to a decision maker

Evolutionary Multi-Objective Optimisation

Try to compute/approximate the Pareto front by EAs



Evolve the population of an EA into a set of Pareto optimal solutions

Approximation algorithms exist for many problems but what about multi-objective optimisation?
→ Many implicit approaches, but no explicit ones!

Preliminaries

We consider minimization problems

- $d \geq 2$ objective dimensions
- objective functions $f_i: S \rightarrow \mathbb{R}$, $1 \leq i \leq d$ map the search space S into the real numbers

Dominance relation

For two objective vectors $x=(x_1, \ldots, x_d)$ and $y=(y_1, \ldots, y_d)$, with
$x, y \in \mathbb{R}^d$, *we define*

$x \preccurlyeq y$ iff $x_i \leq y_i$ for all $1 \leq i \leq d$, (x weakly dominates y)

$x \prec y$ iff $x \preccurlyeq y$ and $x \neq y$. (x strongly dominates y)

Relations translate to search points (elements of S)

Set of all non-dominated objective vectors is called the Pareto front.

Our overall idea for Approximation-Guided Evolution (AGE)

- We keep an unbounded archive A of non-dominated points seen so far.
- The archive is an approximation of the “true” Pareto front.
- The goal is to have a population P that approximates the archive as best as possible.
- We use additive approximation to measure approximation quality.
- Multiplicative approximations can be used in a similar way.

Additive Approximation

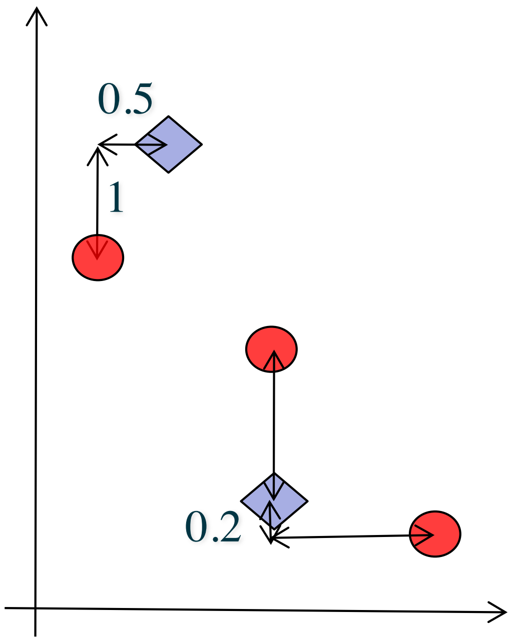
Definition. For finite sets $S, T \subset \mathbb{R}^d$, the additive approximation of T w.r.t. S is defined as

$$\alpha(S,T) := \max_{s \in S} \min_{t \in T} \max_{1 \leq i \leq d} (s_i - t_i)$$

Additive Approximation

Definition. For finite sets $S, T \subset \mathbb{R}^d$, the additive approximation of T w.r.t. S is defined as

$$\alpha(S,T) := \max_{s \in S} \min_{t \in T} \max_{1 \leq i \leq d} (s_i - t_i)$$



Given the set of blue points.
How well does it approximate the red points?

$\alpha(\bullet, \diamond) = \max(1.0; 0.0; 0.2) = 1$

Additive Approximation

Goal. Minimize the approximation of the population P (our output) w.r.t. to the archive A (all points seen so far).

Problem. $\alpha(A, P)$ is not sensitive to local changes of P: measures only improvements of points which are currently worst approximated.

Solution. Consider the set $B=\{\alpha(\{a\}, P) \mid a \in A\}$. Sort B decreasingly and minimize $S_\alpha(A, P) := (\alpha_1, \ldots, \alpha_{|A|})$ lexicographically.

Our contribution

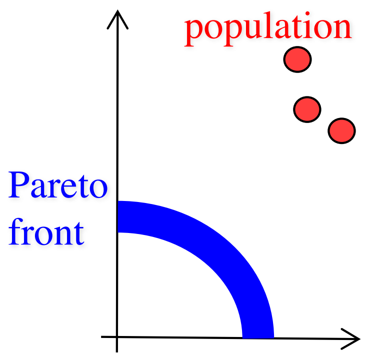
Assuming that the archive approximates the Pareto front, we measure the quality of the population by its approximation w.r.t. the archive:

- Any set of feasible solutions constitutes an approximation of the Pareto front, and
- we optimize the approximation w.r.t. all solutions seen to far.

Our contribution

Assuming that the archive approximates the Pareto front, we measure the quality of the population by its approximation w.r.t. the archive:

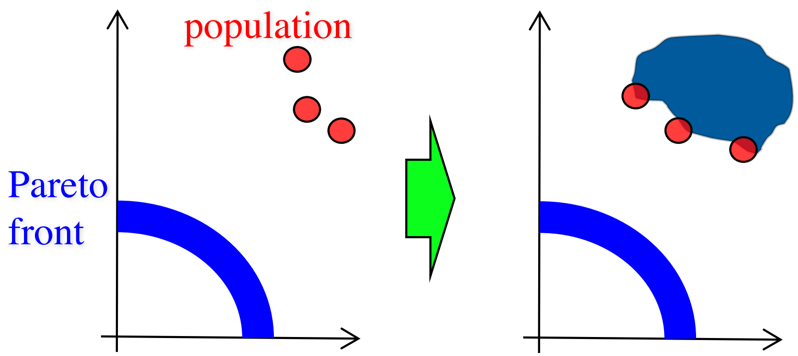
- Any set of feasible solutions constitutes an approximation of the Pareto front, and
- we optimize the approximation w.r.t. all solutions seen to far.



Our contribution

Assuming that the archive approximates the Pareto front, we measure the quality of the population by its approximation w.r.t. the archive:

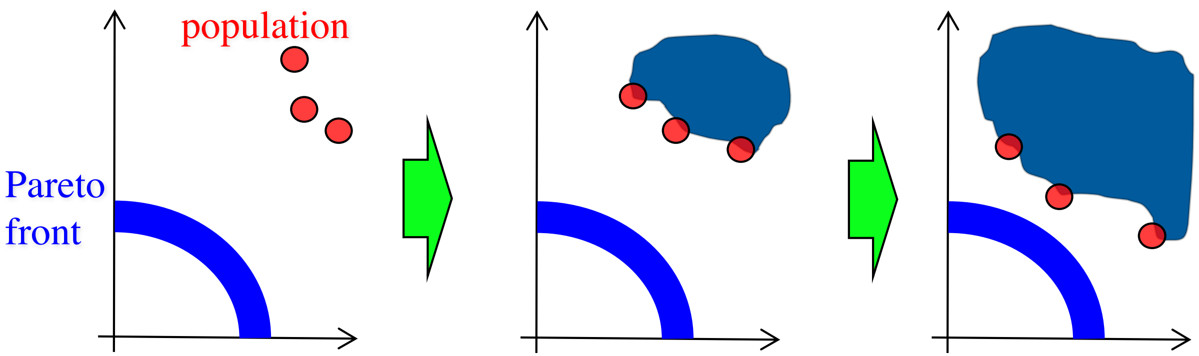
- Any set of feasible solutions constitutes an approximation of the Pareto front, and
- we optimize the approximation w.r.t. all solutions seen to far.



Our contribution

Assuming that the archive approximates the Pareto front, we measure the quality of the population by its approximation w.r.t. the archive:

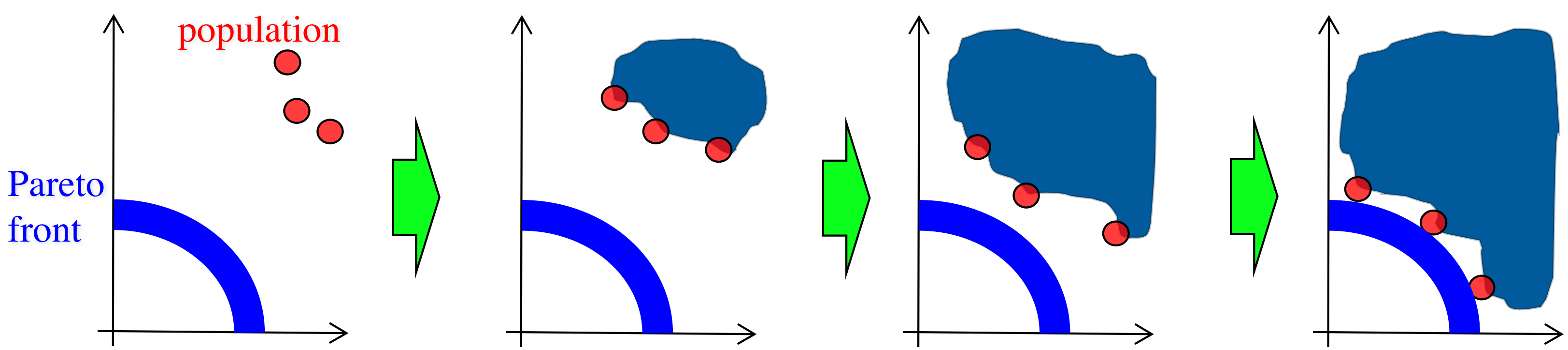
- Any set of feasible solutions constitutes an approximation of the Pareto front, and
- we optimize the approximation w.r.t. all solutions seen to far.



Our contribution

Assuming that the archive approximates the Pareto front, we measure the quality of the population by its approximation w.r.t. the archive:

- Any set of feasible solutions constitutes an approximation of the Pareto front, and
- we optimize the approximation w.r.t. all solutions seen to far.

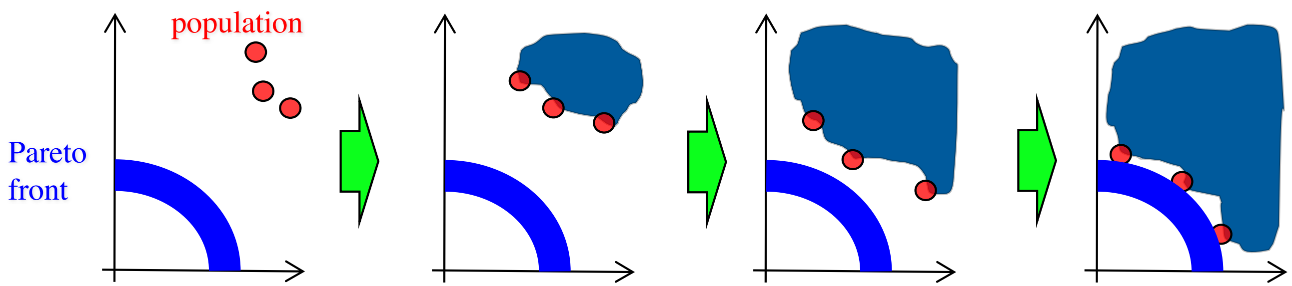


Simple Algorithm

Based on S_α, it is easy to come up with an algorithm!

Population of size μ.

1. Generate λ offspring.
2. Iteratively remove individual p from $(\mu \cup \lambda)$, for which $S_\alpha(A, P \setminus \{ p \})$ is minimal. *drop point with smallest contribution*
3. (Add all non-dominated points to the archive.)



Runtime

We work with a population size of μ *and generate in each generation* λ *offspring.*

Having generated N solutions, we get the following runtime bounds.

Simple algorithm $O(\ N\ (\mu+\lambda)\ |A|\ (\ d\ (\mu+\lambda)\ +\ \log\ |A|\ )\ )$

Works well when $\mu+\lambda$ is small, but e.g. for $\mu+\lambda=100$ becomes slow due to $(\mu+\lambda)^2$ factor.

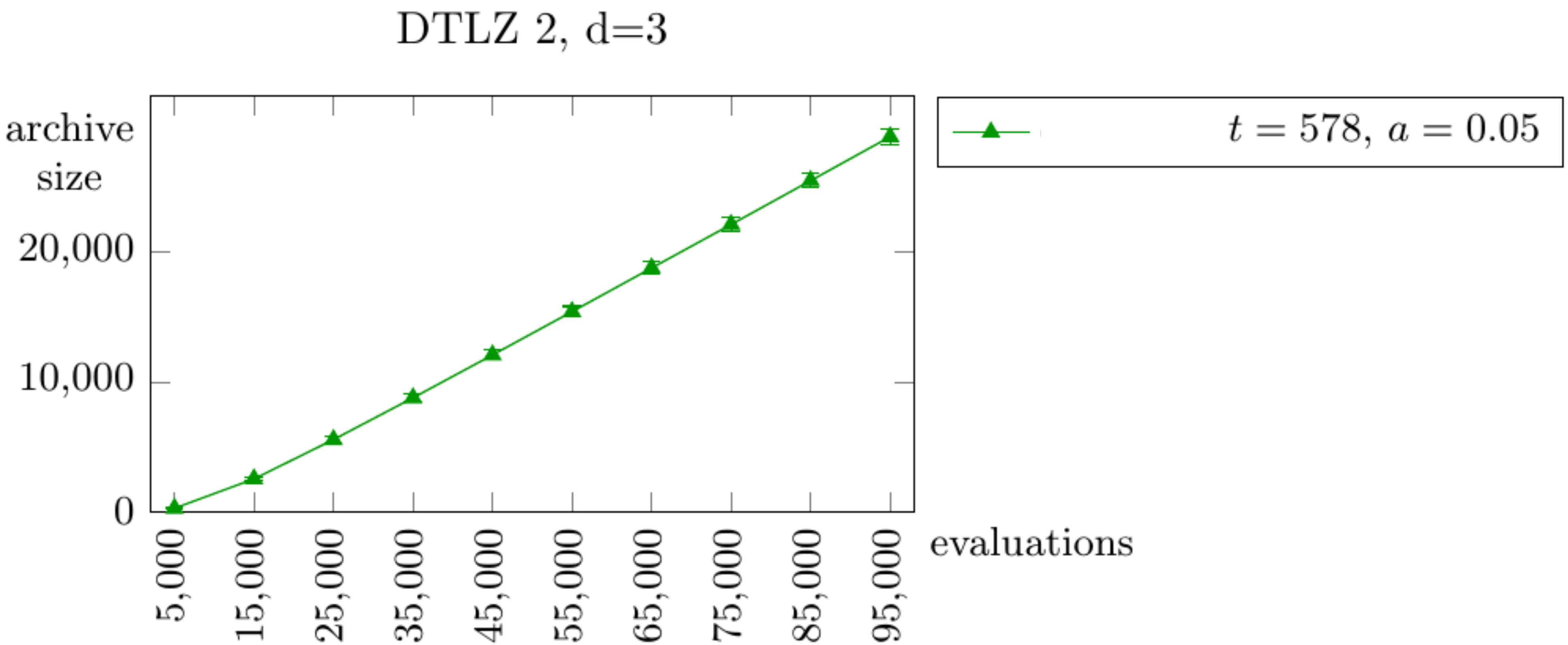
Fast algorithm $O(\ N\ (\mu+\lambda)\ |A| d)$

Idea: clever selection of the μ individuals for the next generation, looking at the worst approximation for which a population point p is responsible.

(for technical details: see IJCAI paper)

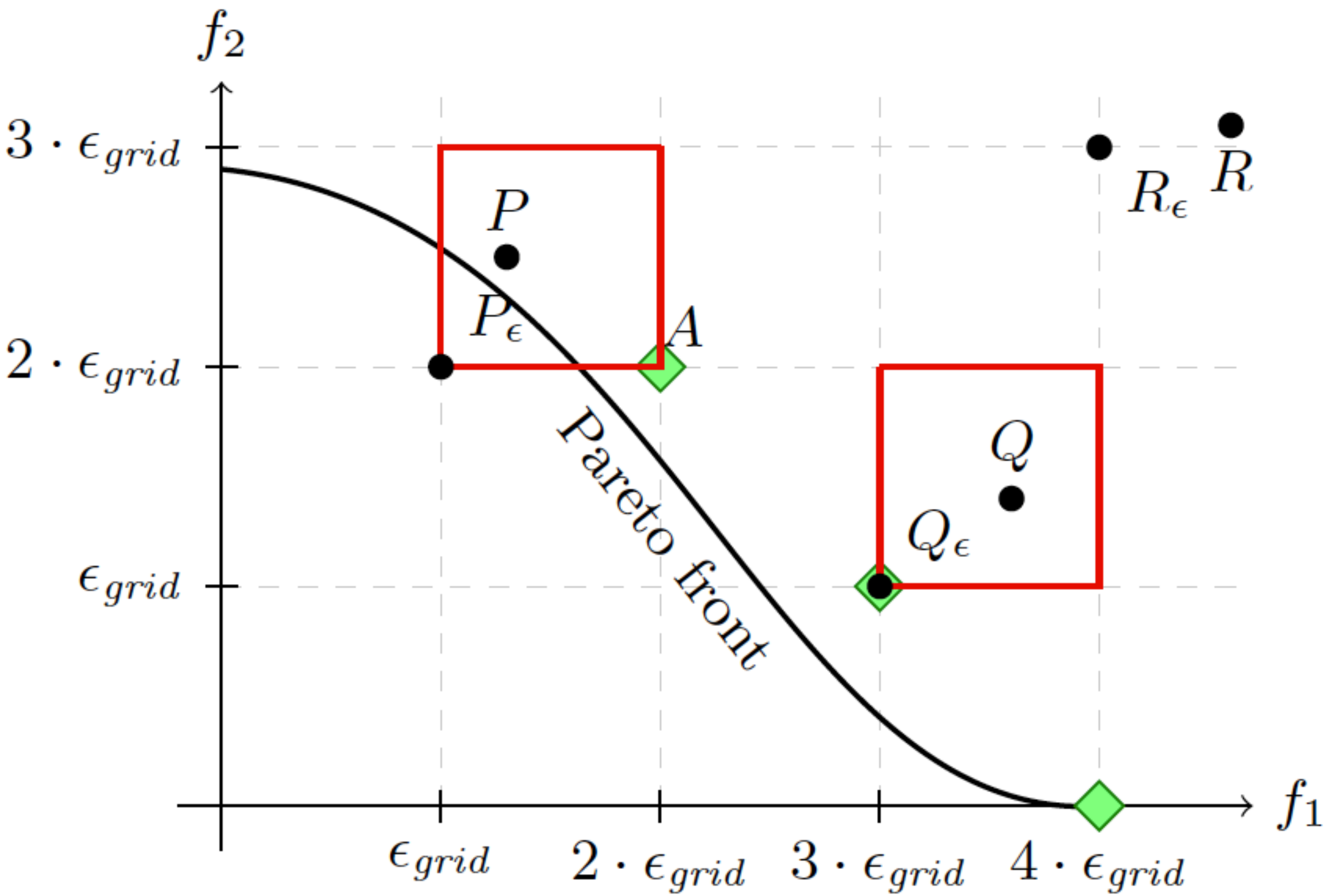
Problem: Runtime grows linearly with the archive size

Development of the Unbounded Archive Size



100.000 evaluations, averages of 100 independent runs

ε-Dominance Approach [based on Laumanns et al. ’02]



- Assign to each objective vector x its box-vector depending of ε_{grid}.

Subroutine 7: Function *floor*

---

input : d-dimensional objective vector x, archive parameter ε_{grid}
output: Corresponding vector v on the ε-grid

1 **for** $i = 1$ **to** d **do** $v[i] \leftarrow \left\lfloor \frac{x[i]}{\varepsilon_{grid}} \right\rfloor$;

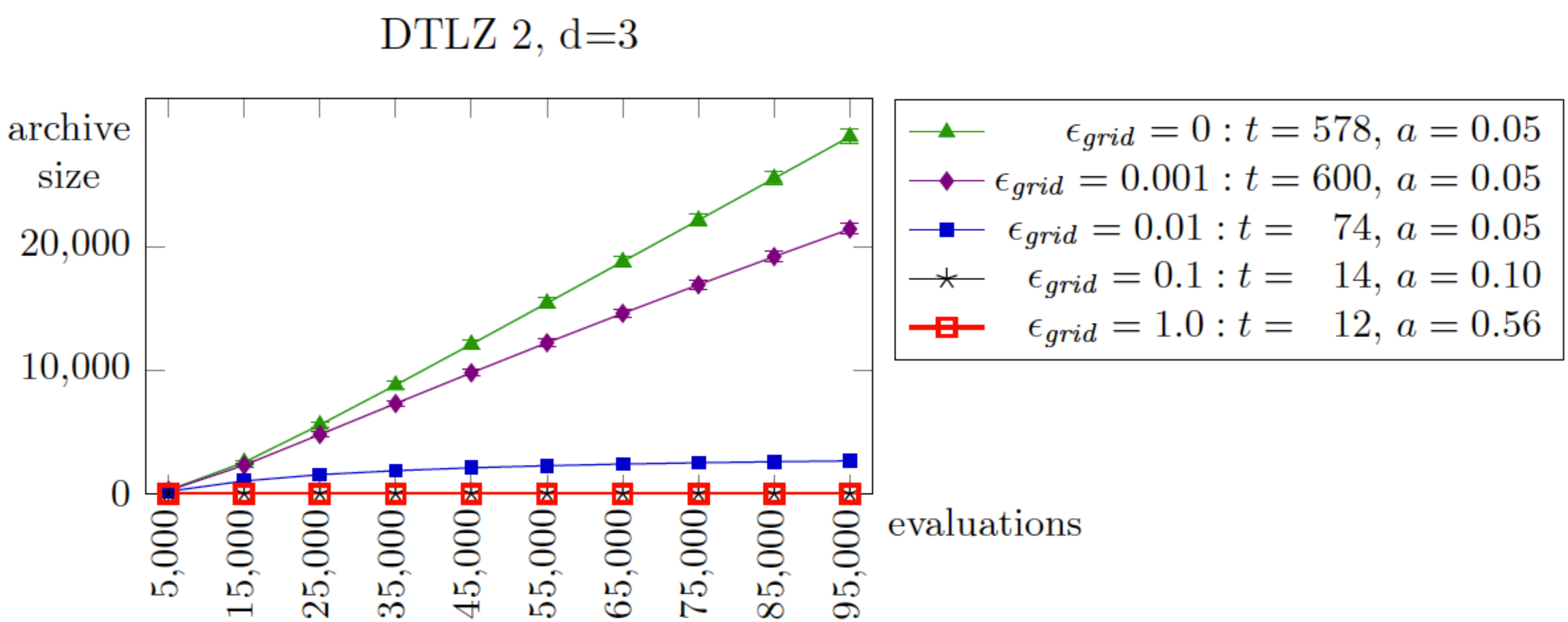
- Archive size is bounded by

$$\left| A^{(t)}_{\varepsilon_{grid}} \right| \leqslant \prod_{j=1}^{d-1} \left\lfloor \frac{K}{\varepsilon_{grid}} \right\rfloor$$

where

$$K = \max_{i=1}^{d} \left(\max_{s \in S} f_i(s) \right)$$

Development of the Archive Size



μ=λ=100.

N=100.000 evaluations, averages of 100 independent runs

Experiments

- NSGA-II, IBEA, SPEA2, SMS-EMOA
  with approx hyp: SMS-EMOA, MO-CMA-ES
  AGE with $\varepsilon_{grid}=0$, $\varepsilon_{grid}=0.1$, $\varepsilon_{grid}=0.01$

Note: MOEA/D was new back then!

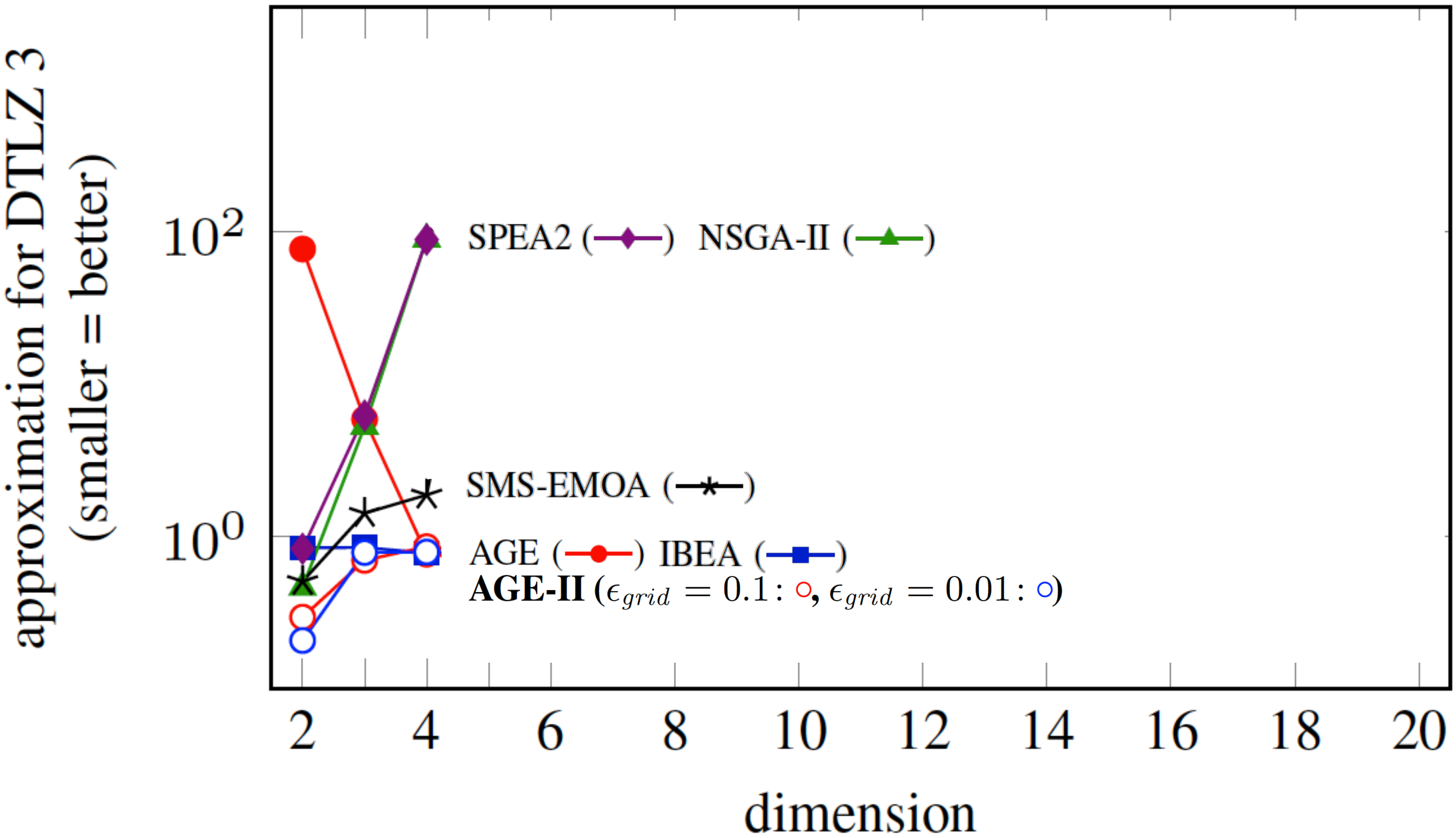
- ZDT 1/2/3/4/6
  WFG 1-9 (each with d=2 and d=3)
  LZ 1-9
  DTLZ 1/2/3/4 (each with d=2,...,20)
  → 80 functions

Limits: 4h (and varying numbers of evaluations)

- μ=100, SBX, PM, implemented in jMetal

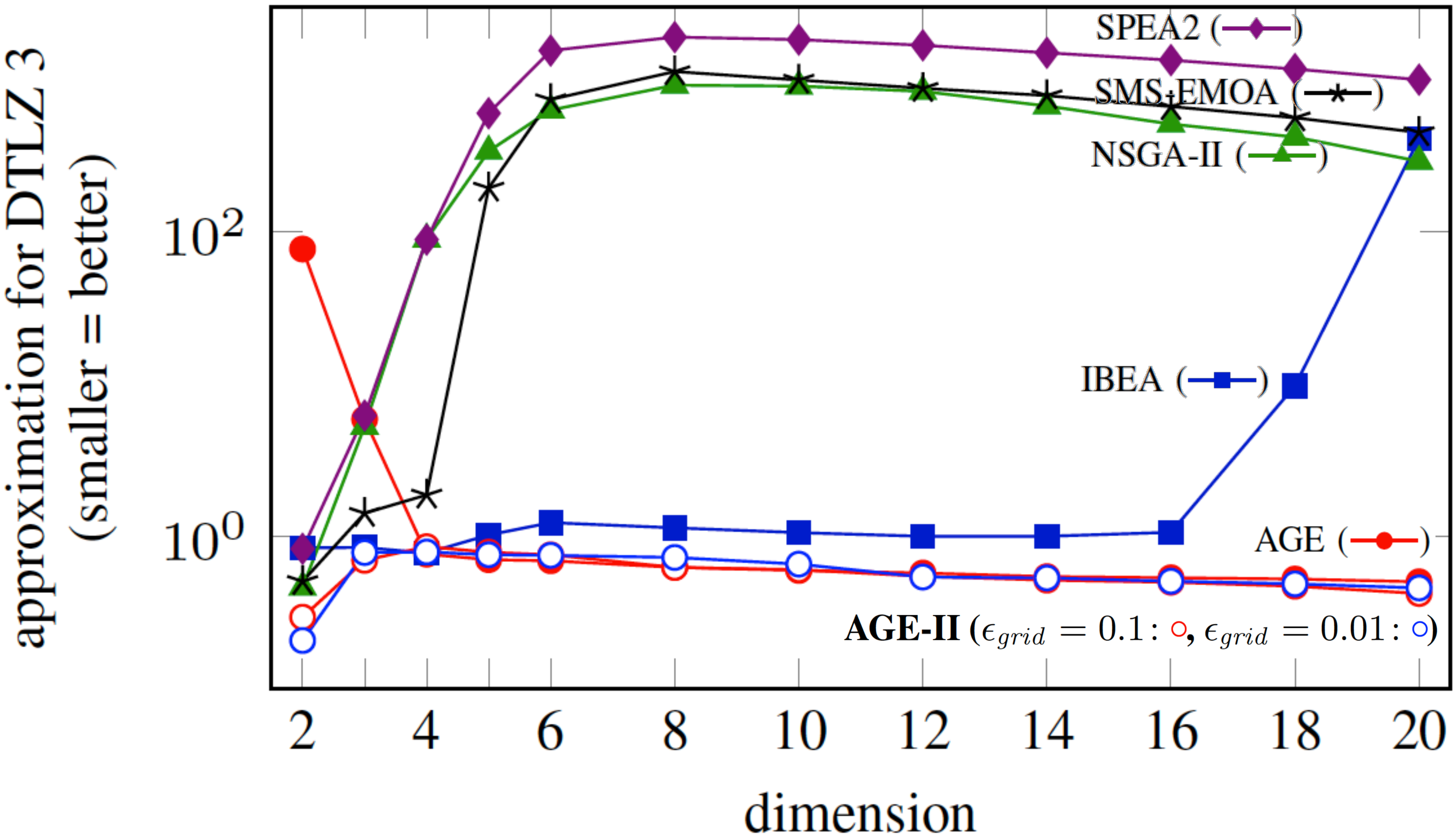
(code is available online:
http://cs.adelaide.edu.au/~markus/publications.html -> GECCO 2013)

Results

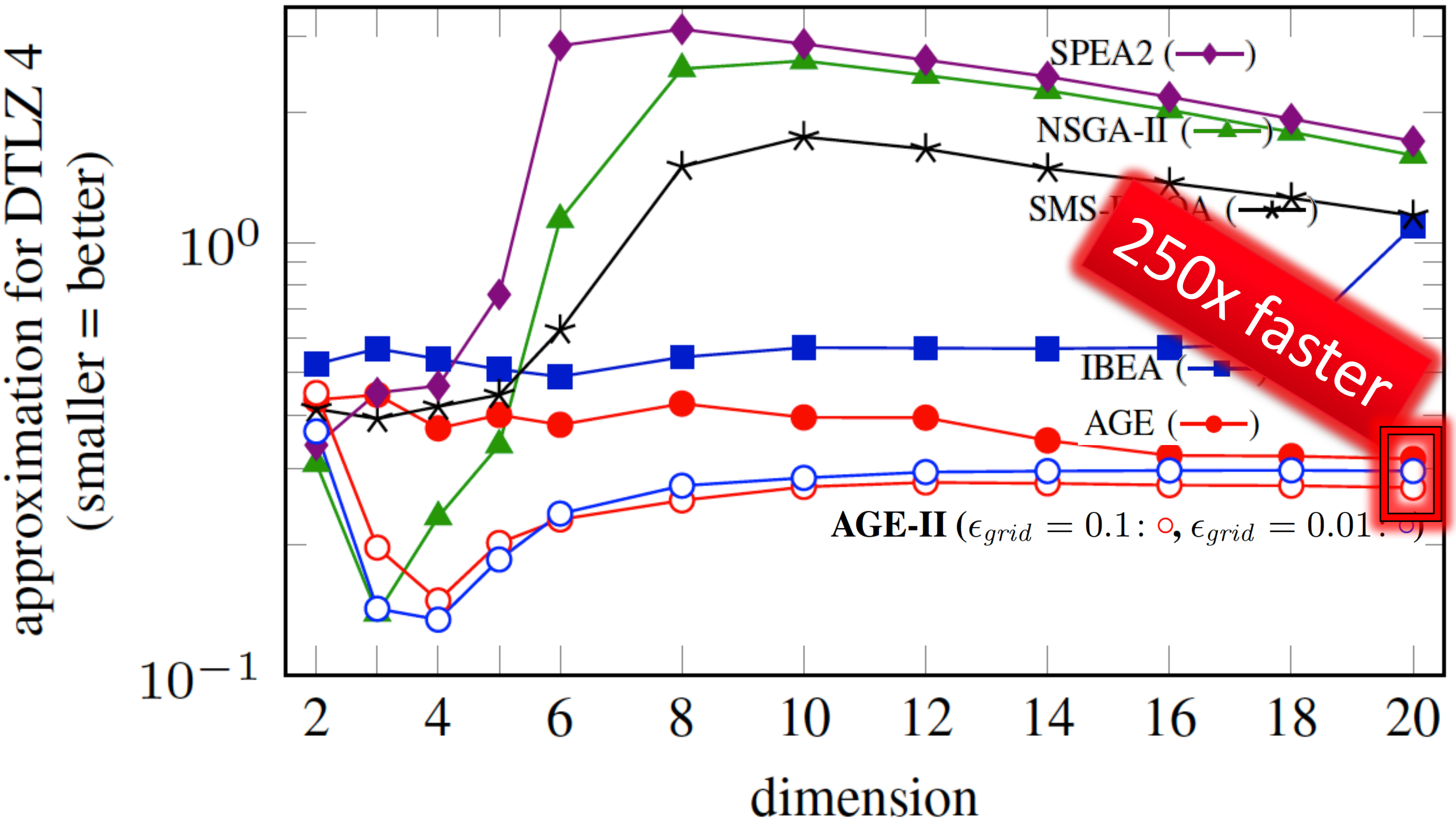


approximation for DTLZ 3
(smaller = better)
10^2
10^0
SPEA2 NSGA-II
SMS-EMOA
AGE IBEA
AGE-II ($\epsilon_{grid} = 0.1$: , $\epsilon_{grid} = 0.01$:)
2
4
6
8
10
12
14
16
18
20
dimension

Results


approximation for DTLZ 3
(smaller = better)
10^2
10^0
2
4
6
8
10
12
14
16
18
20
dimension
SPEA2
SMS-EMOA
NSGA-II
IBEA
AGE
AGE-II ($\epsilon_{grid} = 0.1$: ○, $\epsilon_{grid} = 0.01$: ○)

Results



approximation for DTLZ 4
(smaller = better)
10^0
10^{-1}
2
4
6
8
10
12
14
16
18
20
dimension
SPEA2
NSGA-II
IBEA
AGE
AGE-II ($\epsilon_{grid} = 0.1$: ○, $\epsilon_{grid} = 0.01$: ○)
250x faster

Summary

- Approximated Guided Evolution (AGE) for multi-objective optimization which works with a formal notion of additive/multiplicative approximation.

- AGE outperforms state-of-the-art approaches, in terms of additive approximation and covered hypervolume (for DTLZ 1 and 3), given a fixed time budget (4h).

- This holds, in particular, for problems with many objectives, which most other algorithms have difficulties dealing with.

EMO Applications (in Adelaide)

- Team Cycling: race time vs energy consumption
- Android Apps: energy consumption vs deviation from test oracle
- Wind energy: power output vs area vs cable
- Wave energy: power output vs area vs cable
- Travelling thief: profit vs weight collected

...and others...

Note: typically, our code is online.

Travelling Thief Problem

http://cs.adelaide.edu.au/~optlog/research/

With code, instances, results, papers, … (two competitions)

THE UNIVERSITY OF ADELAIDE AUSTRALIA

ALBERT-LUDWIGS-UNIVERSITÄT FREIBURG

南京航空航天大学 Nanjing University of Aeronautics & Astronautics

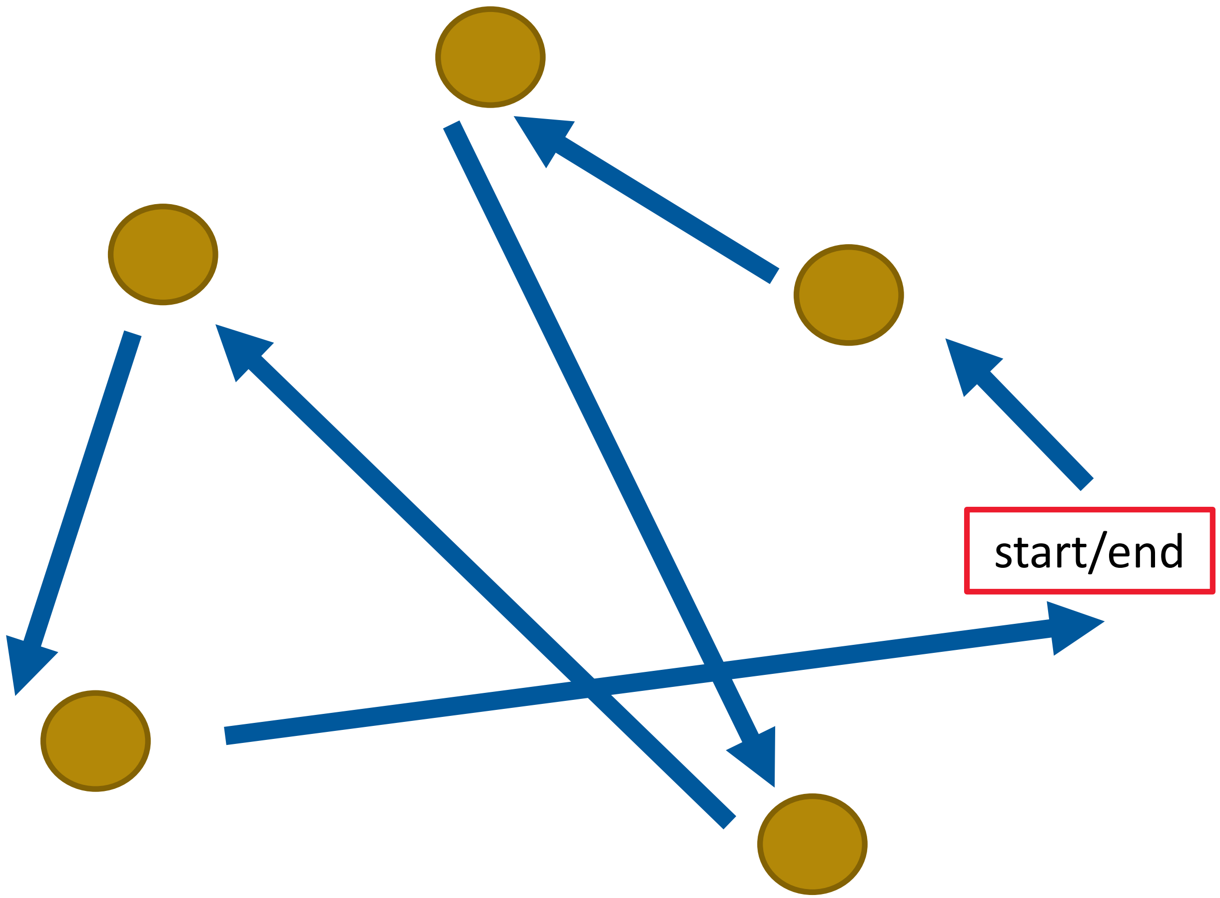
The University Of Sheffield.

A case study of algorithm selection for the travelling thief problem

Joint work with: Marius Lindauer, Mustafa Mısır, Samadhi Nallaperuma, Frank Hutter

Travelling Thief Problem (2013, read-world characteristic: interdependent problems)

Definition
TSP part: *n* cities



Travelling Thief Problem (2013, read-world characteristic: interdependent problems)

Definition

TSP part: *n* cities

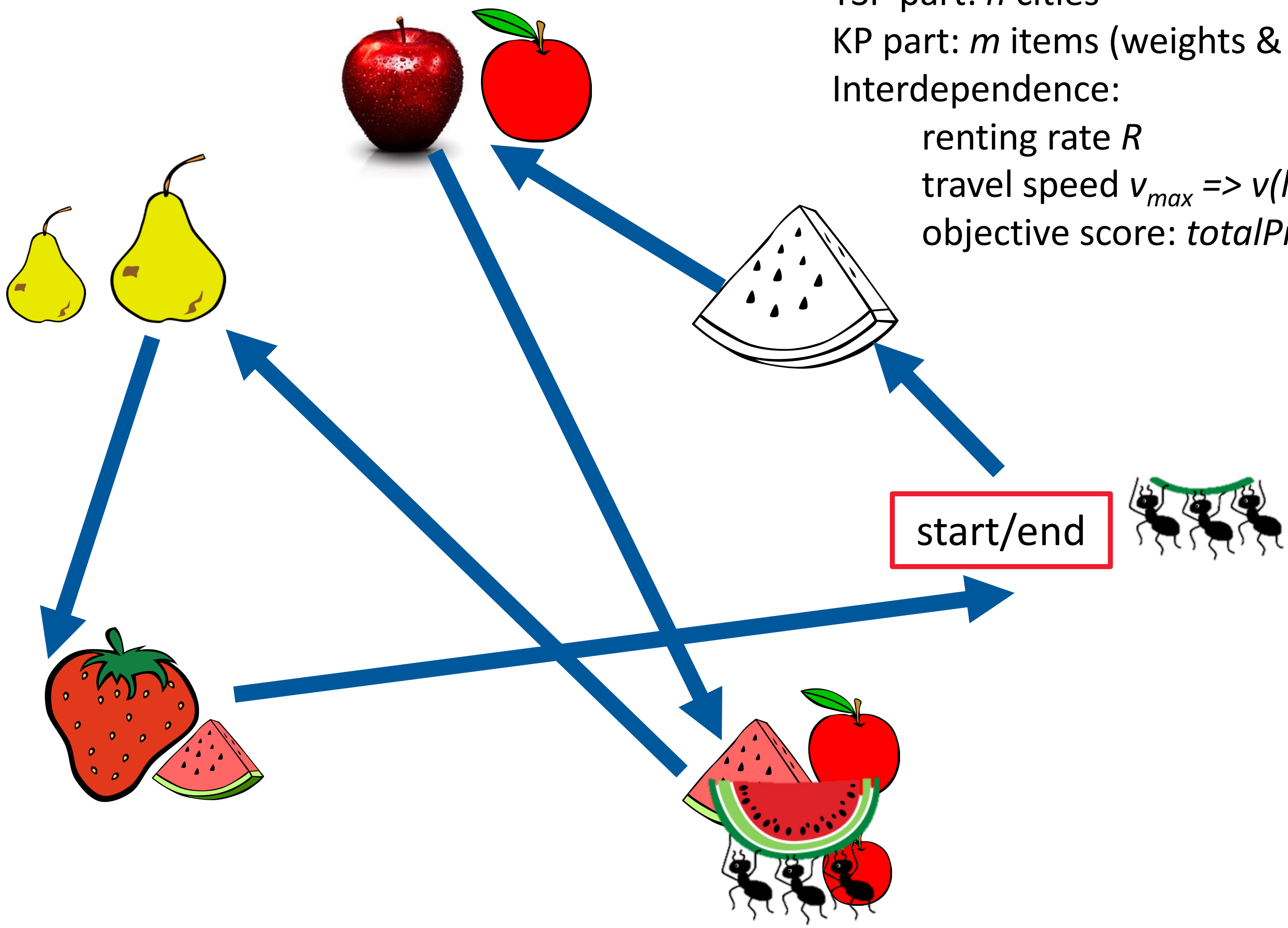
KP part: *m* items (weights & profits), capacity *W*

Interdependence:

renting rate *R*

travel speed v_{max} => *v(load)* => v_{min}

objective score: *totalProfit – R * travelTime*



Renting rate

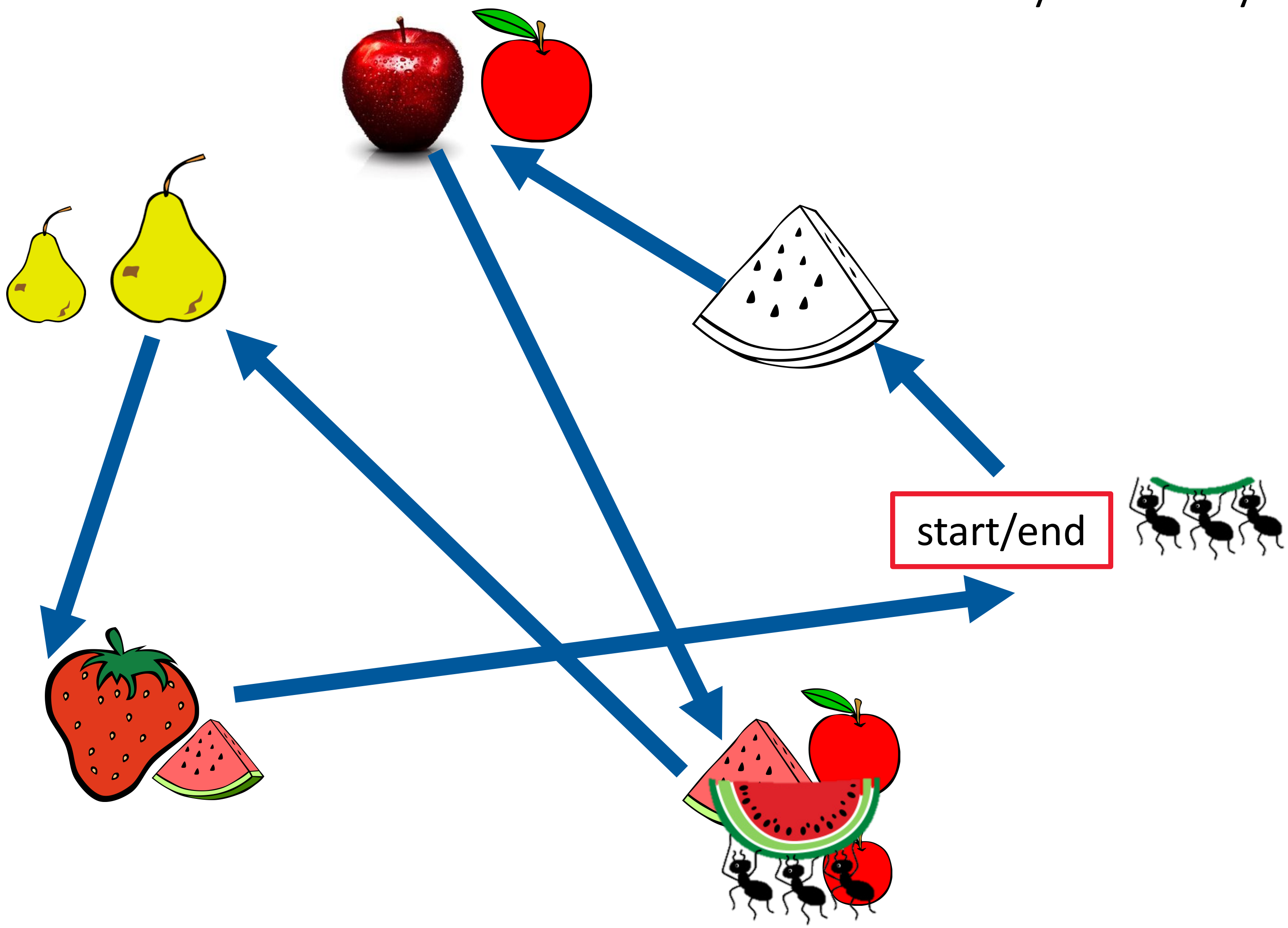
Travel from city i to i+1 in π

$$Z([\Pi, P]) = \sum_{i=1}^{n} \sum_{k=1}^{m_i} p_{ik} y_{ik} - R \left(\frac{d_{x_n x_1}}{\upsilon_{max} - \nu W_{x_n}} + \sum_{i=1}^{n-1} \frac{d_{x_i x_{i+1}}}{\upsilon_{max} - \nu W_{x_i}} \right)$$

profits

Travel from last city to first city

$$\nu = \frac{\upsilon_{max} - \upsilon_{min}}{W}$$



TTP Situation (2016)

- Many algorithms have been introduced:
  - Initially generic hill-climbers, successively more and more understanding was encoded
  - Deterministic construction heuristics, restart strategies, holistic approaches
  - MIP & dynamic programming for special case
  - Increasing computational cost
- "Best algorithm" depends on instance (given the computation budget of 10 minutes).

- There are exact approaches based on Dynamic Programming now, I might get to them later…

Our Contributions

1. Comprehensive dataset for algorithm performance comparison
   (21 algorithms on 9720 instances)
2. Comprehensive dataset for instance analysis
   (55 features of 9720 instances)
3. Algorithm portfolios based on 1. and 2.
4. Analysis of 3.

1. Algorithm Performance

- History
  - Bonyadi et al. (2013): 4 cities, 6 items, exhaustive enumeration
  - Polyakovskiy et al. (2014):
    - 9720 instances established with up to almost 100k cities and 1m items
    - First heuristics:
      1. Strong focus on very good TSP tours (using LKH).
      2. Packing plan creation using hill-climbers or a deterministic construction heuristic.
  - Since then: more construction heuristics, co-evolutionary approaches, holistic attempts, fast implementations of search operators (for quick objective score update), special case algorithms, …

1. Algorithm Performance

- 9720 Instances vary widely
  - 51-85,900 cities (based on TSPlib)
  - three different KP types (shown to have different difficulties for KP solvers)
  - 1-10 items per city, different KP sizes
  - Renting rate R set so that there is at least on TTP solution with objScore=0=opt(KP)−R*opt(TSP)
- Researchers use not all of them (except Polyakovskiy et al., 2014), for example:
  - Mei et al. (2014): 30 instances with 11k-34k cities
  - Faulkner et al. (2015): 72 instances with 195 to 86k cities
  - Wagner (2016): 108 instances with 51 to 1000 cities

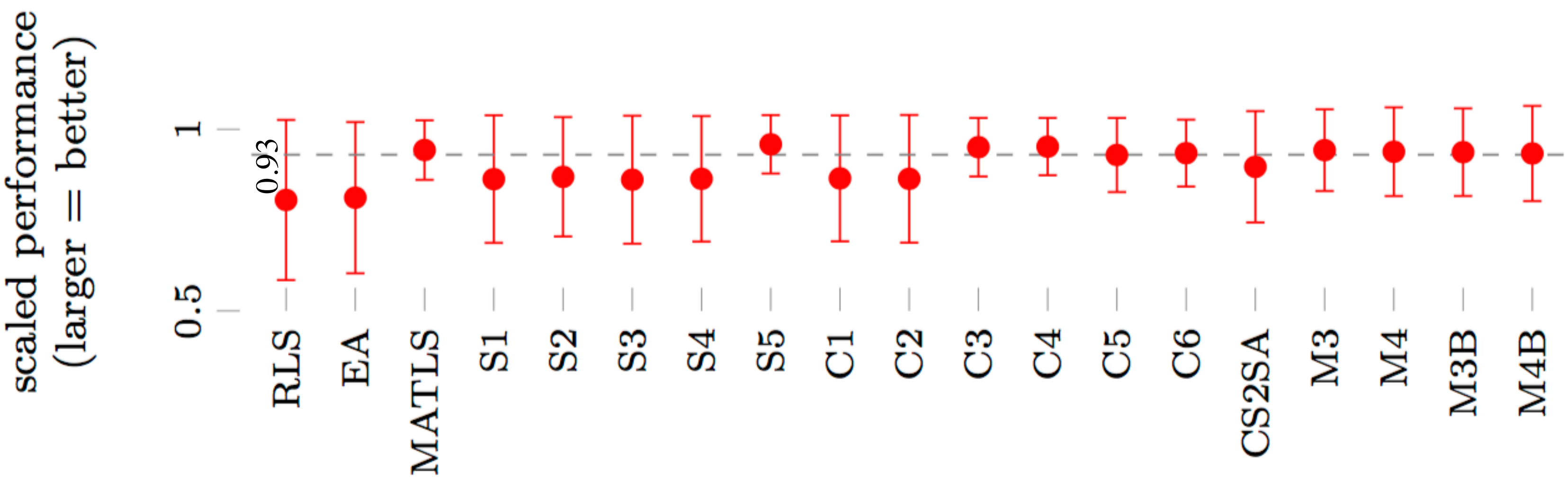
→ complete picture not possible

1. Algorithm Performance

- Benchmarking:
  - 9720 instances, once, 10 minutes, rescaled to [0,1], -1 for crash/time-out
  - Averages over all:

worst average performance

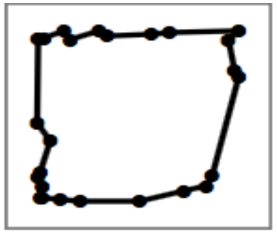
best ever performance

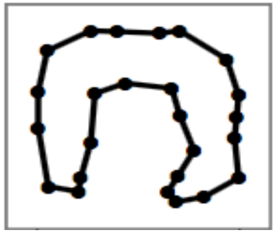


Construction heuristics SH/DH left out due to poor performance.
Dataset available online: http://tinyurl.com/ttpadelaide

2. Instance Characteristics

- 47 TSP features (Mersmann et al. 2012/2013, Nallaperuma et al. 2013/2014, ...)
  - 11 distance features (min/max/mean/fractions/...)
  - 1 mode feature (distribution of edge cost)
  - 6 cluster features (GDBSCAN, number of clusters, mean distances to cluster centroids)
  - 6 nearest neighbour features (min/max/mean/...)
  - 5 angle features (min/... between node an NN)
  - 11 MST features (min/max/mean depth, ...)
  - 2 convex hull features





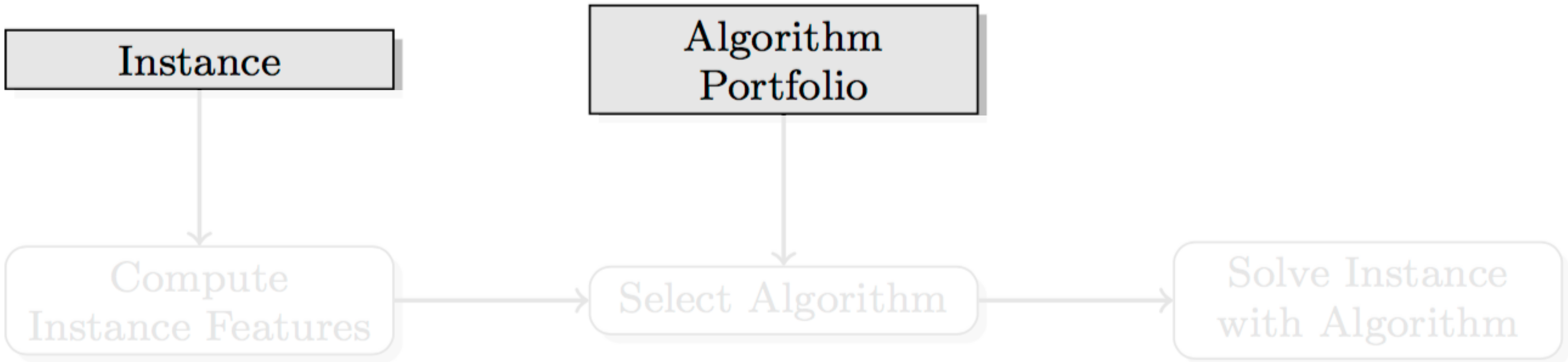
- 4 KP features
  - Capacity, knapsack type, total number of items, number of items per city
- TSP: number of cities
- 3 TTP features
  - Renting ratio, minimum travel speed, maximum travel speed

Note: not too many are "really" TTP-specific.

Dataset available online: http://tinyurl.com/ttpadelaide

3. Algorithm Selection

- As seen previously: no single algorithm dominates all others on all instances.
- Exploit this using algorithm selection (idea from the 1970s).



- Major success story SATzilla (2008): empirical performance model predicts performance of an algorithm and selects the one with best prediction + schedule to solve easy instances without instance feature overhead

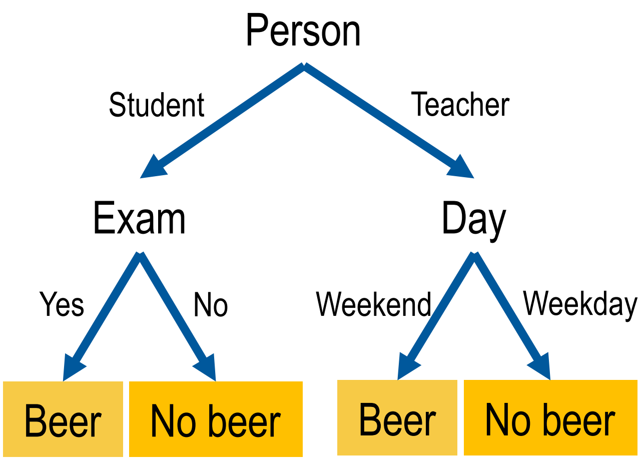
3. Algorithm Selection

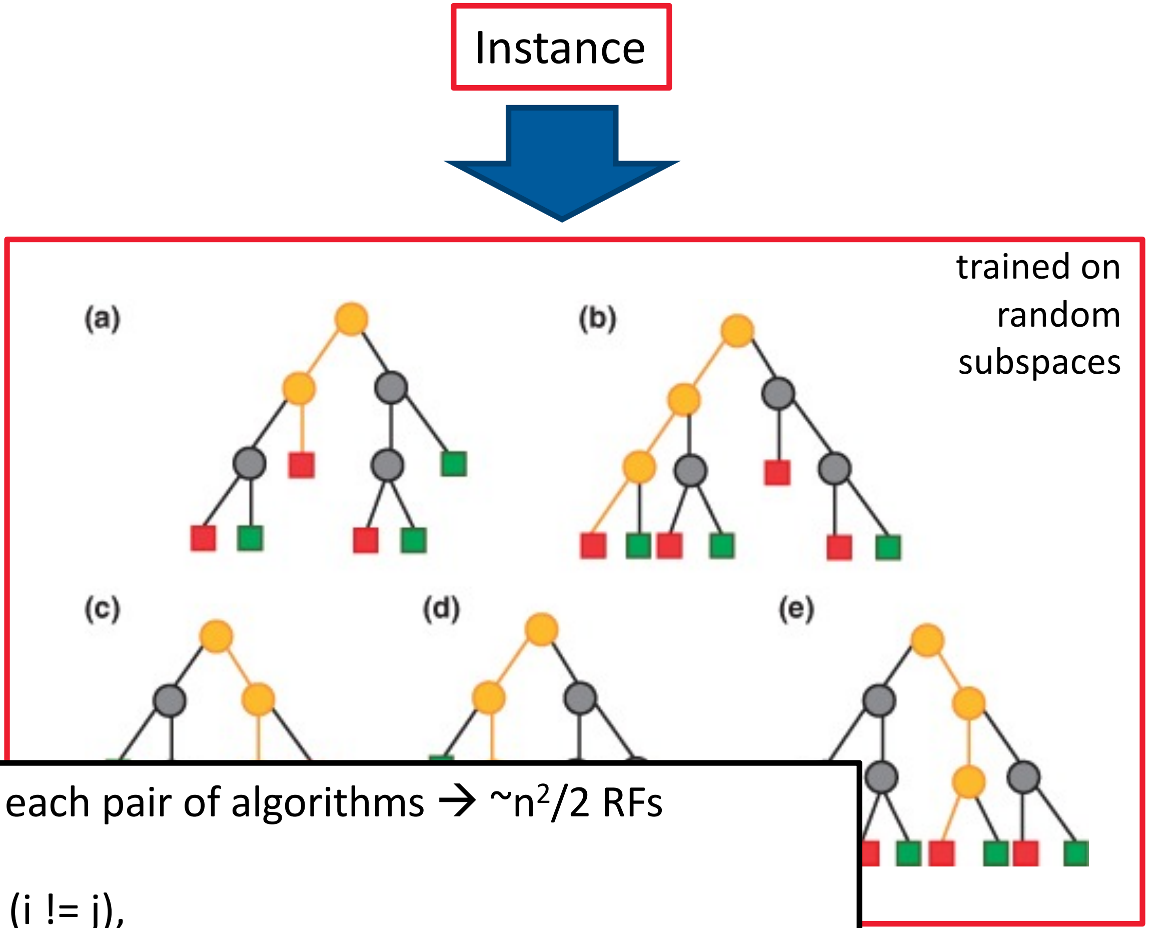
- We are using AutoFolio (Lindauer et al. 2015):
  - FlexFolio (Hoos et al. 2014): several different algorithm selection methods
  - SMAC (Hutter et al. 2011): search for best selection approach + parameter tuning

- Example: AutoFolio determines whether classification or regression performs better, and in case of classification the parameters of a random forest (many decision trees) are tuned.

[Random Forest]

Random forest: lots of decision trees





Actually, there is an RF with 10 trees for each pair of algorithms → ~$n^2/2$ RFs

Then, for each pair of algorithms <A_i,A_j> (i != j),
a random forests (consisting of 10 randomized decision trees) votes either for A_i or A_j and this one gets a point.
In the end, the algorithms with the most points gets selected.

Output: 4x red, 1x green → red

https://www.researchgate.net/publication/260436143_Machine_learning_methods_in_chemoinformatics/figures?lo=1

3. Algorithm Selection

- Results

Simulated System	Approach	Performance
Single Best (S5)	Baseline	**0.959**
Oracle	Theoretical Optimum	1.0
SATzilla'09-like	Regression (Lasso-Regression)	0.966
SATzilla'11-like	Pairwise Classification (RF)	0.993
ISAC-like	Clustering (k-means)	0.989
3S-like	Classification (k-NN)	0.992

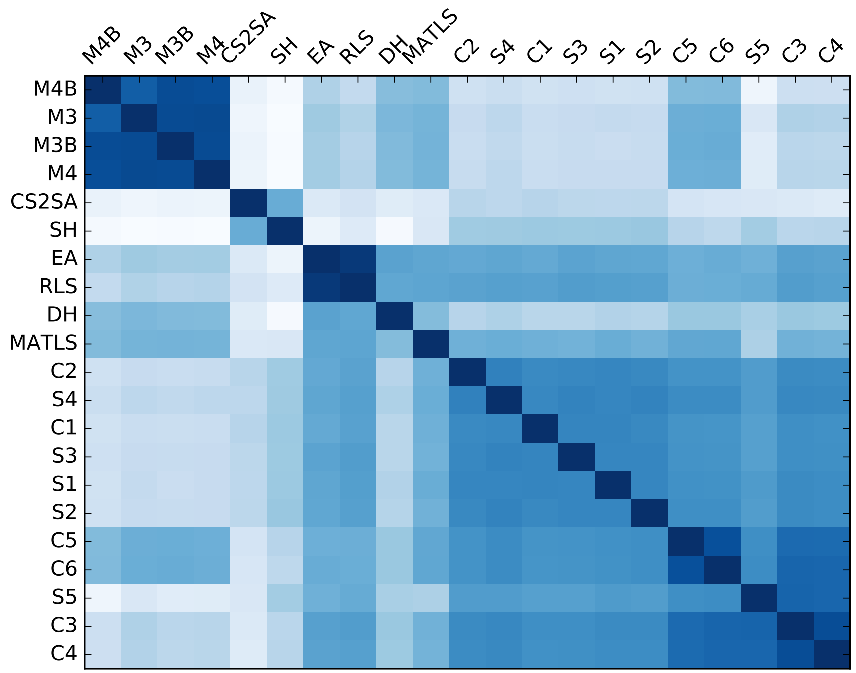
Comparing different algorithm selection approaches on TTP

Near-1 performance might be due to the large number of instances (almost 10k).

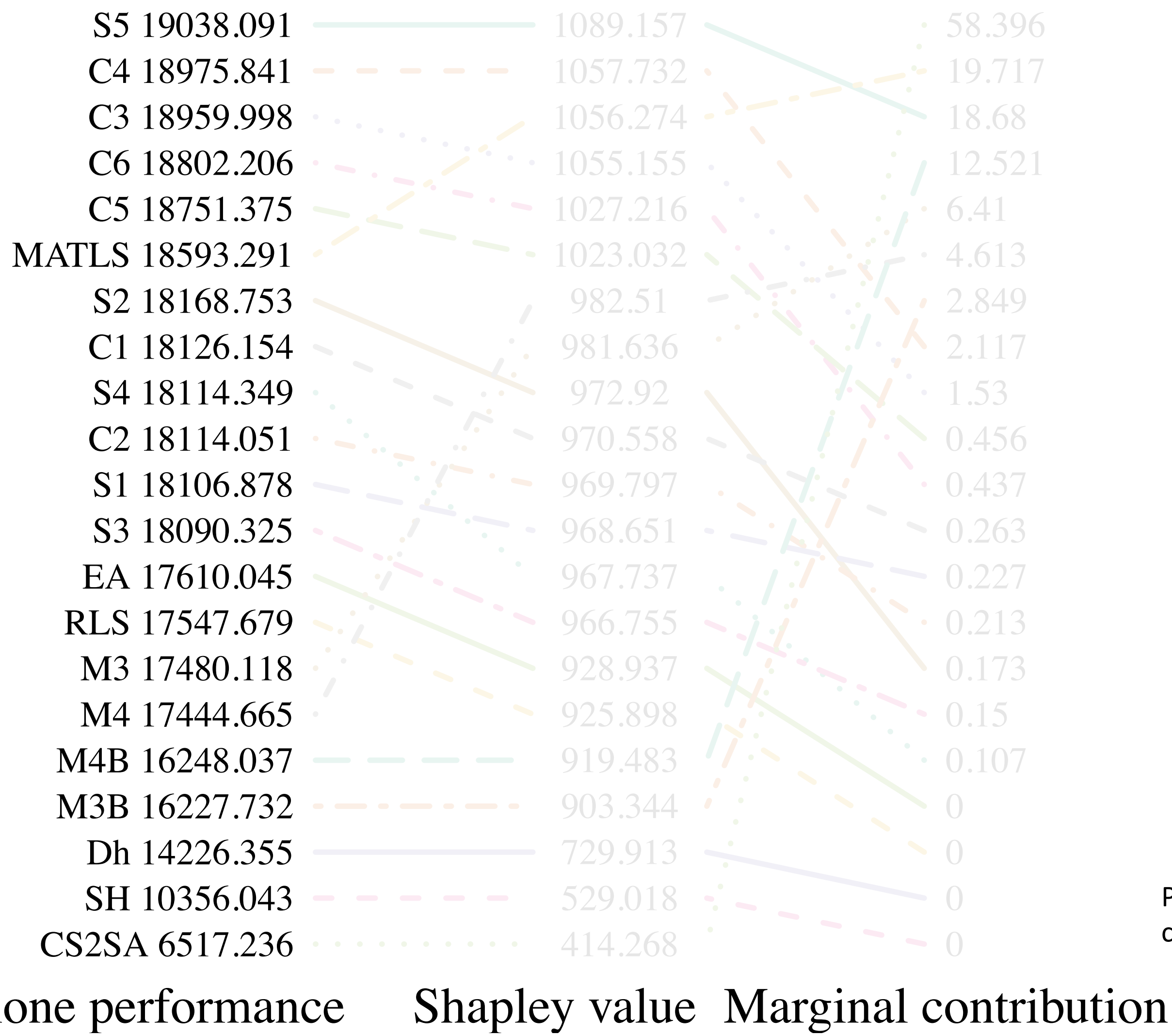
AutoFolio (1d, 4 cores) vs Satzilla'11-like: negligible improvement (chose RF, tuned parameters).

4. Portfolio Analysis

- Complementarity important for good portfolios
  - Single best vs oracle: difference of only 0.041
  - Remember that 19 of 21 algorithms had >0.8 avg.
- Correlations across instances (Spearman's rank coefficients), and clustered
  - Algorithms form clusters reflecting their historical development
  - Analysis of similarity only (not performance)



4. Portfolio Analysis



Problem: too much credit for similar algorithms, fails to consider synergies

(sum across all instances, +9720 offset for negative performance)

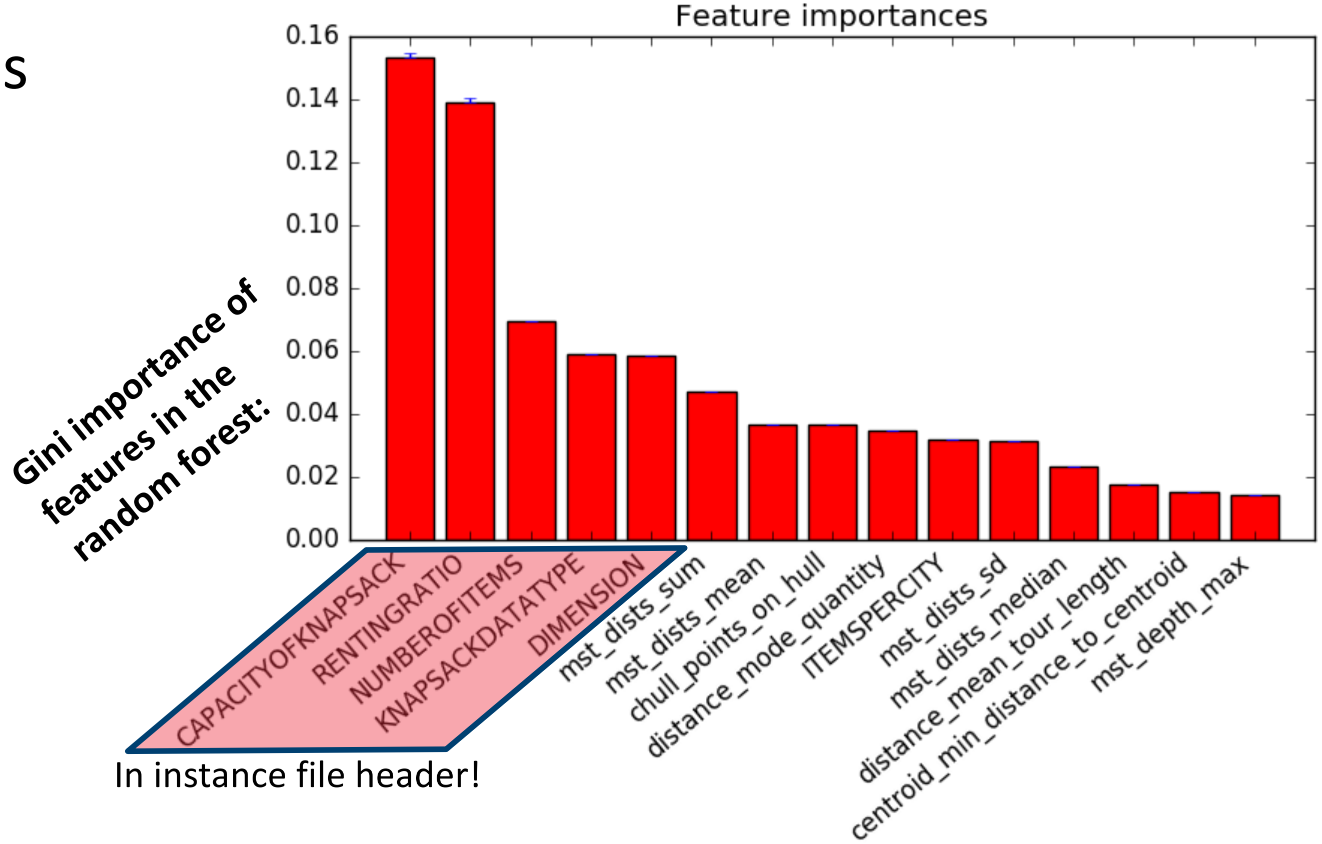
(contribution to any subset of the algorithm portfolio)

(performance increase of portfolio when algorithm is added)

Problem: penalises correlated algorithms

4. Portfolio Analysis

Feature calculation times need to be considered (e.g. almost 10 minutes for pla7397* instances)



How about portfolios that use only Top 1-5 features? →
(S5 only: **0.959**, best portfolio before: **0.993**)
Top 1-5: **0.977**, 0.980, 0.986, 0.988, **0.992**

4. Portfolio Analysis

- What else did we learn?
  - Challenging: lots of dimensions to navigate, 10k instances, 21 algorithms, noise in the underlying algorithm performance data
  - For example, using only KP capacity:
    - The smallest 1/3rd of the instances is dominated by the most complex algorithms, amongst those the ones that produce solutions with the longest tours.
    - The largest 1/3rd is dominated by CS2SA (a fast implementation of search operators) and S5 (resampling solutions).
    - Algorithm selection in the central 1/3rd seems to be difficult. (why?)

→ Certain algorithms dominate, but they are not very complementary as only few feature values are necessary to achieve near-optimal portfolio performance.

Summary

- New datasets established:
  - 21 algorithms on 9720 instances
  - Raw data available as CSV and in the ASlib format http://cs.adelaide.edu.au/~optlog/research/ttp.php, ASlib URL to be added
- Portfolios:
  - Few algorithms needed
  - Few features needed (can be determined quickly)
- Future directions:
  - Representative subset (which criteria?)
  - More analyses
  - (more algorithms…)

http://cs.adelaide.edu.au/~markus/

The slides will be made available today.

Markus Wagner

markus.wagner@adelaide.edu.au

”Packing While Travelling”

- Simplification of the TTP
- Tour is fixed, and we only deal with the packing component
- Sergey/Frank: DP/FPTAS
- This gave rise to the first non-trivial complete TTP approach (SEAL 2017), for relatively small instances

Traveling Thief Problem (TTP)

Renting rate

Travel from city i to i+1 in π

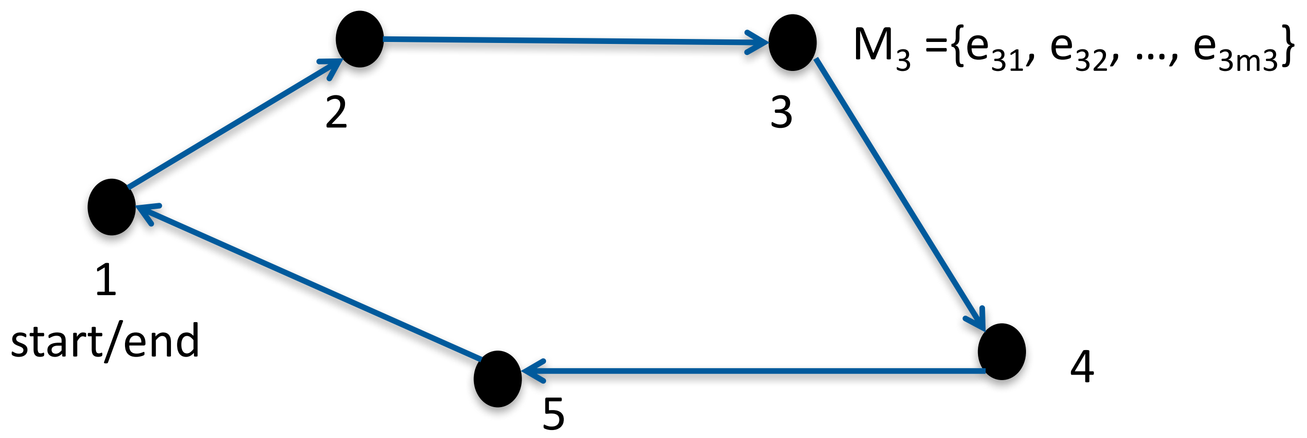
- Fitness is given by

$$Z([\Pi, P]) = \sum_{i=1}^{n} \sum_{k=1}^{m_i} p_{ik} y_{ik} - R\left(\frac{d_{x_n x_1}}{\upsilon_{max} - \nu W_{x_n}} + \sum_{i=1}^{n-1} \frac{d_{x_i x_{i+1}}}{\upsilon_{max} - \nu W_{x_i}}\right)$$

$$\nu = \frac{\upsilon_{max} - \upsilon_{min}}{W}$$

profits

Travel from last city to first city



Packing While Traveling

Assume that the tour is fixed . Then we only have to deal with the packing component.

$$B(x) = P(x) - R \cdot T(x),$$

where

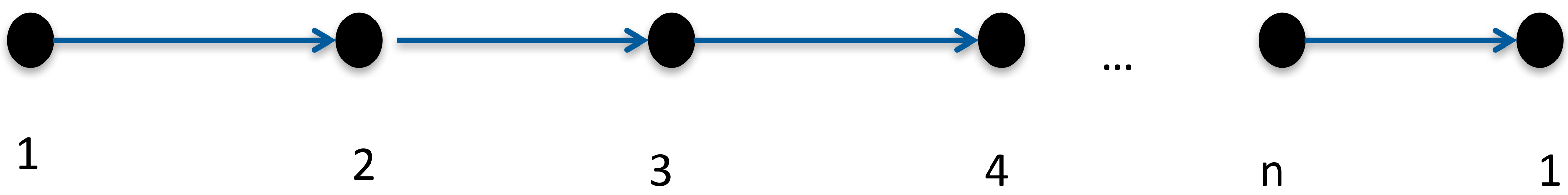
profit

Cost dependent on weight of chosen items

$$P(x) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} p_{ij} x_{ij}$$

represents the total profit of selected items and

$$T(x) = \sum_{i=1}^{n} \frac{d_i}{\upsilon_{max} - \nu \sum_{k=1}^{i} \sum_{j=1}^{m_k} w_{kj} x_{kj}}$$



Dynamic Programming for PWT

- Sort the items as they appear on the path, breaking ties for items at the same city arbitrarily.
- Use dynamic programming (similar to classical 0/1 knapsack) and process the items in sorted order. Store for the first i items and each possible weight the maximal possible benefit (delete dominated entries).
- Size of the table in polynomial in m and the maximum possible weight => algorithm with pseudopolynomial runtime.

DP for PWT

Store for the first i cities on the path and every possible weight, the maximal possible objective value.

$$\max\begin{cases}\beta_{i',j',k}\\ \beta_{i',j',k-w_{ij}}+p_{ij}-Rd_{in}\left(\frac{1}{v_{max}-\nu k}-\frac{1}{v_{max}-\nu(k-w_{ij})}\right)\end{cases}$$

	[1]	[2]					[n]
weight							
k-w_{ij}		$B_{i',j'}$					150
							200
k		$B_{i',j'}$	$B_{i,j}$				210
							220
							210

To decide: keep the previous plan OR add the item?

Experimental Results (Exact)

Instance	m	OPT	Exact Approaches		
			eMIP	BIB	DP
			RT(s)	RT(s)	RT(s)
uncorr_01	100	1651.697	1.217	5.694	0.027
uncorr_06	100	10155.4942	12.605	3.698	0.065
uncorr_10	100	10297.7134	3.525	0.795	0.036
uncorr-s-w_01	100	2152.6188	0.328	7.566	0.001
uncorr-s-w_06	100	4333.8512	12.59	2.215	0.012
uncorr-s-w_10	100	9048.4908	37.144	1.107	0.022
b-s-corr_01	100	4441.9852	1.42	125.954	0.014
b-s-corr_06	100	10260.9767	4.509	22.541	0.101
b-s-corr_10	100	13630.6153	11.013	27.081	0.187
uncorr_01	500	17608.5781	19.594	27.581	0.247
uncorr_06	500	56294.5239	384.213	13.354	2.829
uncorr_10	500	66141.484	211.302	2.325	4.01
uncorr-s-w_01	500	13418.8406	4.337	34.866	0.09
uncorr-s-w_06	500	34280.473	346.43	7.285	1.04
uncorr-s-w_10	500	50836.6588	519.902	3.338	2.022
b-s-corr_01	500	21306.9158	40.482	624.204	1.534
b-s-corr_06	500	69370.2367	236.387	97.313	14.616
b-s-corr_10	500	82033.9452	376.569	218.728	22.011
uncorr_01	1000	36170.9109	218.306	114.567	1.872
uncorr_06	1000	93949.1981	1261.949	36.847	20.944
uncorr_10	1000	122963.6617	620.896	4.821	30.116
uncorr-s-w_01	1000	27800.9614	241.957	399.158	0.802
uncorr-s-w_06	1000	61764.4599	1152.624	12.792	9.872
uncorr-s-w_10	1000	103572.4074	2146.408	7.644	15.047
b-s-corr_01	1000	46886.1094	378.551	6129.531	11.783
b-s-corr_06	1000	125830.6887	643.533	919.201	94.523
b-s-corr_10	1000	161990.5015	862.572	1646.52	151.601

NP-hardness (Non-negative benefit)

- PWT solutions can attain positive and negative values.

Theorem 2. *Given a PWT instance, the problem to decide whether there is a solution x with $B(x) \geq 0$ is NP-complete.*

- This rules out meaningful multiplicative approximations.

FPTAS for PWT

- Let $B(\emptyset) = -R \cdot \sum_{i=1}^{n} d_i / v_{\max}$

  the baseline travel cost when the vehicle travels empty.
- Consider the objective function $B'(x) = B(x) - B(\emptyset)$

  which gives the amount gained over the baseline travel cost.
- Let $OPT = \max_{x \in \{0,1\}^m} B'(x).$
- We design a fully polynomial time approximation scheme for B'. Solution x of quality $B'(x) \geq (1-\epsilon)OPT.$

  Runtime polynomial in n and 1/ε.

FPTAS for PWT

- Assume each item e_{ij}, on its own makes a positive contribution.
- Considering the single items e_{ij}, we have.

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i}(P(e_{ij}) - R \cdot T(e_{ij}))x^*_{ij} - B(\emptyset) \geq B(x^*) - B(\emptyset) = OPT$$

- Pick item with the largest value B' value and set $L = max_{e_{ij} \in M} B'(e_{ij}) > 0$
- We have $L \geq OPT/m$ and $L \leq OPT$.
- Set $r = \epsilon L/m$, round $B'(x)$ to $\lfloor (B'(x)/r \rfloor$ and run DP.
- Number of rows in DP table is upper bounded by

$$(OPT/r) + 1 \leq OPT/(\epsilon L/m) + 1 \leq m^2/\epsilon + 1$$

- Error in each step is at most $r = \epsilon L/m \leq \epsilon OPT/m$
- At most m steps. So, we get $B'(x) \geq (1-\epsilon)OPT$.

Algorithm 1 FPTAS for $B'(x)$

- Set $L = \max_{e_{ij} \in M} B'(e_{ij})$, $r = \epsilon L/m$, and $d_{in} = \sum_{l=i}^{n} d_l$, $1 \leq i \leq n$.
- Compute order $\preceq$ on the items e_{ij} by sorting them in lexicographic order with respect to their indices (i, j).
- For the first item e_{ij} according to $\preceq$, set $\beta(i, j, 0) = B'(\emptyset)$ and $\beta(i, j, w_{ij}) = B'(e_{ij})$.
- Consider the remaining items of M in the order of $\preceq$ and do for each item e_{ij} and its predecessor $e_{i'j'}$:
  - In increasing order of k do for each $\beta(i', j', k)$ with $\beta(i', j', k) \neq -\infty$
    - If there is no $\beta(i, j, k')$ with ($\lfloor \beta(i, j, k')/r \rfloor \geq \lfloor \beta(i', j', k)/r \rfloor$ and $k' < k$),
      set $\beta(i, j, k) = max\{\beta(i, j, k), \beta(i', j', k)\}$.
    - If there is no $\beta(i, j, k')$ with ($\lfloor \beta(i, j, k')/r \rfloor \geq \lfloor \beta(i', j', k + w_{ij})/r \rfloor$ and $k' < k + w_{ij}$),
      set $\beta(i, j, k + w_{ij}) = max\{\beta(i, j, k + w_{ij}), \beta(i', j', k) + p_{ij} + Rd_{in}(\frac{1}{v_{\max} - \nu k} - \frac{1}{v_{max} - \nu(k + w_{ij})})\}$.

Theorem 3. *Algorithm 1 is a fully polynomial time approximation scheme (FPTAS) for the objective B'. It obtains for any ϵ, $0 < \epsilon \leq 1$, a solution x with $B'(x) \geq (1 - \epsilon) \cdot OPT$ in time $O(m^3/\epsilon)$.*

Experiments FPTAS

Instance	m	DP		FPTAS													
				$\epsilon = 0.0001$		$\epsilon = 0.001$		$\epsilon = 0.01$		$\epsilon = 0.1$		$\epsilon = 0.25$		$\epsilon = 0.5$		$\epsilon = 0.75$	
		OPT	RT(s)	AR(%)	RT(s)	AR(%)	RT(s)	AR(%)	RT(s)	AR(%)	RT(s)	AR(%)	RT(s)	AR(%)	RT(s)	AR(%)	RT(s)
instance family `eil101_large-range`																	
uncorr_01	100	69802802.2801	0.03	100	0.002	100	0.002	100	0.002	100	0.002	100	0.002	100	0.002	100	0.029
uncorr_06	100	204813765.6933	0.053	100	0.019	100	0.02	100	0.019	100	0.019	100	0.019	100	0.019	100	0.049
uncorr_10	100	172176182.1249	0.041	100	0.028	100	0.028	100	0.028	100	0.028	100	0.027	100	0.026	99.9628	0.037
uncorr-s-w_01	100	36420530.5753	0.006	100	0.003	100	0.003	100	0.003	100	0.003	100	0.003	100	0.002	100	0.004
uncorr-s-w_06	100	148058928.2952	0.098	100	0.072	100	0.502	100	0.072	100	0.069	100	0.065	100	0.059	100	0.07
uncorr-s-w_10	100	142538516.4602	0.136	100	0.101	100	0.104	100	0.103	99.9978	0.096	99.9978	0.086	99.9978	0.073	99.9978	0.089
m-s-corr_01	100	19549602.2671	0.003	100	0.002	100	0.002	100	0.002	100	0.002	100	0.002	100	0.001	100	0.002
m-s-corr_06	100	137203175.1921	0.147	100	0.115	100	0.118	100	0.113	100	0.089	100	0.063	100	0.04	100	0.043
m-s-corr_10	100	225584278.6004	0.424	100	0.326	100	0.329	100	0.312	100	0.2	100	0.179	100	0.086	100	0.073
uncorr_01	500	385692662.0930	0.47	100	0.451	100	0.454	100	0.619	100	0.508	100	0.445	100	0.43	100	0.517
uncorr_06	500	958013934.6172	3.539	100	3.749	100	7.431	100	3.947	100	3.69	99.9996	3.677	99.9996	3.486	99.9993	3.021
uncorr_10	500	844949838.4389	4.87	100	5.393	100	5.716	100	5.483	100	5.135	100	4.851	99.9992	4.609	99.9992	4.295
uncorr-s-w_01	500	182418888.9364	1.157	100	1.157	100	1.199	100	1.145	99.9995	1.112	99.9995	1.063	99.9995	0.977	99.9904	0.929
uncorr-s-w_06	500	780432253.0187	22.39	100	25.04	100	26.276	100	24.024	100	23.282	99.9997	21.756	99.9997	18.293	99.9997	18.411
uncorr-s-w_10	500	714433353.7957	30.959	100	34.458	100	39.004	100	34.308	100	32.308	99.9996	28.792	99.999	26.392	99.999	25.971
m-s-corr_01	500	96463941.1275	2.335	100	2.478	100	2.782	100	2.695	100	1.509	100	0.963	100	0.546	100	0.408
m-s-corr_06	500	666701000.1488	108.705	100	126.833	100	139.63	100	122.75	100	62.479	100	33.547	100	17.959	100	10.642
m-s-corr_10	500	1082009880.5886	262.999	100	299.862	100	317.352	100	274.284	100	145.087	100	78.47	99.9994	41.816	99.9994	25.924
uncorr_01	1000	777386336.9660	4.222	100	4.397	100	4.347	100	4.309	100	4.341	100	4.377	100	4.28	100	4.24
uncorr_06	1000	1933319297.4248	46.043	100	51.383	100	53.087	100	48.861	100	52.957	99.9999	52.062	99.9997	50.286	99.9996	51.488
uncorr_10	1000	1693797490.1704	64.485	100	76.744	100	78.847	100	74.128	100	82.754	100	77.057	100	72.283	100	72.567
uncorr-s-w_01	1000	361991311.8336	14.254	100	15.072	100	15.67	100	14.523	100	14.11	100	14.039	100	12.088	100	11.129
uncorr-s-w_06	1000	1574469459.3163	286.843	100	318.096	100	330.508	100	337.289	100	334.318	100	307.588	99.9998	270.013	99.9996	245.927
uncorr-s-w_10	1000	1439410696.3695	393.793	100	438.775	100	455.83	100	464.527	100	441.955	100	433.672	99.9994	378.917	99.9994	340.813
m-s-corr_01	1000	191170309.5684	46.858	100	58.031	100	59.987	100	58.101	100	31.703	100	18.771	100	10.728	100	6.831
m-s-corr_06	1000	1315708161.7720	2393.205	100	2512.281	100	2606.412	100	1921.573	100	666.749	100	364.452	100	208.969	100	150.06
m-s-corr_10	1000	2163713055.3759	6761.49	100	6668.535	100	6441.906	100	4526.653	100	1334.882	100	703.258	100	397.527	100	282.211

DP for TTP

- Let $[\Pi, f(\cdot)]$ be the best solution obtained when using permutation π
- We can obtain an optimal solution for TTP by considering all permutations, $Z^* = \arg\max_{\forall \Pi, w \in \cdot} Z([\Pi, f(w)])$

Idea:

- Adapt dynamic programming for TSP to TTP by making use of DP for PWT.
- Let S be a subset of nodes and 1 be the first city of the tour.
- The DP for TSP stores for each S and endpoint k, the shortest path from city 1 to city k visiting all cities in S exactly once at [S,k].
- For TTP store at [S,k,w] the largest benefit when ending at city k with weight w (and visiting all cities in S exactly once)

DP for TTP

- Let $\dot{S} = N \setminus \{1\}$ all cities except the first one.
- Let $\overline{W}_{x_n}$ and $\overline{P}_{x_n}$ be the total weight and profit of items picked at city x_n. We have

$$\begin{aligned} Z([\dot{S}, 1, f_{x_n}(W_{x_n})]^*) = Z([\dot{S} \setminus \{x_n\}, x_n, f_{x_{n-1}}(W_{x_n} - \overline{W}_{x_n})]) \\ + \overline{P}_{x_n} - R\left(\frac{d_{x_n x_1}}{v_{max} - \nu W_{x_n}}\right). \end{aligned}$$

- In general, we can compute $[S, i, f_j(W_j)]$ from

$$[S \setminus \{j\}, j, f_{j-1}(W_j - \overline{W}_j)], \text{ where } i \in \dot{S} \setminus S \text{ and } j \in S.$$

- Compute entries for each of the 2^n subsets and n-1 endpoints.

Experiments TTP (Exact)

Instance	n	m	Running time (sec.) DP	BnB	CP
eil51_n05_m4_uncorr_01	5	4	0.018	0.023	0.222
eil51_n06_m5_uncorr_01	6	5	0.07	0.079	0.24
eil51_n07_m6_uncorr_01	7	6	0.143	0.195	0.497
eil51_n08_m7_uncorr_01	8	7	0.343	0.505	4.594
eil51_n09_m8_uncorr_01	9	8	0.633	1.492	63.838
eil51_n10_m9_uncorr_01	10	9	0.933	5.188	776.55
eil51_n11_m10_uncorr_01	11	10	2.414	23.106	12861.181
eil51_n12_m11_uncorr_01	12	11	3.938	204.786	-
eil51_n13_m12_uncorr_01	13	12	14.217	2007.074	-
eil51_n14_m13_uncorr_01	14	13	13.408	36944.146	-
eil51_n15_m14_uncorr_01	15	14	89.461	-	-
eil51_n16_m15_uncorr_01	16	15	59.526	-	-
eil51_n17_m16_uncorr_01	17	16	134.905	-	-
eil51_n18_m17_uncorr_01	18	17	366.082	-	-
eil51_n19_m18_uncorr_01	19	18	830.18	-	-
eil51_n20_m19_uncorr_01	20	19	2456.873	-	-

Evaluation Heuristics on small benchmarks

	TTP-DP		MA2B			C5		DP-S5	
Instance	OPT	RT	Gap	Std	RT	Gap	Std	Gap	Std
eil51_n05_m4_multiple-strongly-corr_01	619.227	0.02	29.1	12.1	2.71	35.5	1.20e-6	41.3	0.0
eil51_n05_m4_uncorr_01	466.929	0.02	0.0	0.0	3.22	0.0	2.20e-6	0.0	2.20e-6
eil51_n05_m4_uncorr-similar-weights_01	299.281	0.02	0.0	0.0	3.21	7.8	2.40e-6	7.8	1.20e-6
eil51_n05_m20_multiple-strongly-corr_01	773.573	0.08	13.4	0.0	1.44	14.3	0.0	12.8	0.0
eil51_n05_m20_uncorr_01	2144.796	0.07	0.0	0.0	3.35	7.4	0.0	6.6	2.30e-6
eil51_n05_m20_uncorr-similar-weights_01	269.015	0.04	0.0	0.0	3.51	0.0	2.30e-6	0.0	0.0
eil51_n10_m9_multiple-strongly-corr_01	573.897	1.21	0.0	0.0	6.07	0.0	0.0	0.0	0.0
eil51_n10_m9_uncorr_01	1125.715	0.93	0.0	0.0	6.06	0.0	1.30e-6	0.0	1.30e-6
eil51_n10_m9_uncorr-similar-weights_01	753.230	0.86	0.0	0.0	5.87	0.0	0.0	0.0	0.0
eil51_n10_m45_multiple-strongly-corr_01	1091.127	14.89	0.0	0.0	7.99	0.0	0.0	0.0	0.0
eil51_n10_m45_uncorr_01	6009.431	6.39	0.0	0.0	8.6	6.6	2.30e-6	0.0	0.0
eil51_n10_m45_uncorr-similar-weights_01	3009.553	8.87	0.0	0.0	6.78	0.0	2.30e-6	0.0	2.30e-6
eil51_n12_m11_multiple-strongly-corr_01	648.546	4.58	0.0	0.0	6.08	4.6	2.20e-6	4.6	2.20e-6
eil51_n12_m11_uncorr_01	1717.699	3.94	0.0	0.0	7.21	0.0	1.20e-6	0.0	1.20e-6
eil51_n12_m11_uncorr-similar-weights_01	774.107	3.36	0.0	0.0	7.03	0.0	2.30e-6	0.0	2.30e-6
eil51_n12_m55_multiple-strongly-corr_01	1251.780	117.99	0.0	0.0	9.19	0.0	0.0	0.0	0.0
eil51_n12_m55_uncorr_01	8838.012	35.79	0.0	0.0	9.76	0.0	0.0	0.0	0.0
eil51_n12_m55_uncorr-similar-weights_01	3734.895	38.36	12.3	0.0	8.34	12.3	0.0	0.2	0.0
eil51_n15_m14_multiple-strongly-corr_01	547.419	39.82	0.0	0.0	7.87	14.1	1.30e-6	13.3	1.30e-6
eil51_n15_m14_uncorr_01	2392.996	89.46	0.0	0.0	7.28	3.8	0.0	3.8	0.0
eil51_n15_m14_uncorr-similar-weights_01	637.419	16.35	0.0	0.0	6.86	0.0	1.60e-6	0.0	1.60e-6
eil51_n15_m70_multiple-strongly-corr_01	920.372	3984.29	2.1	1.1	12.11	0.0	2.70e-6	0.0	2.70e-6
eil51_n15_m70_uncorr_01	9922.137	740.22	0.0	0.0	9.67	7	1.20e-6	1.9	0.0
eil51_n15_m70_uncorr-similar-weights_01	4659.623	867.78	0.0	0.0	7.98	0.0	0.0	0.0	0.0
eil51_n16_m15_multiple-strongly-corr_01	794.745	105.5	0.0	0.0	7.7	18.9	1.6e-6	18.9	1.6e-6
eil51_n16_m15_multiple-strongly-corr_10	4498.848	623.4	0.0	0.0	9.1	12.9	0.0	16.6	1.3e-6
eil51_n16_m15_uncorr_01	2490.889	59.5	1.0	0.7	8.4	1.6	2.3e-6	1.6	2.3e-6
eil51_n16_m15_uncorr_10	3601.077	211.5	0.0	0.0	9.0	7.1	1.6e-6	7.1	1.6e-6
eil51_n16_m15_uncorr-similar-weights_01	540.897	36.4	0.0	0.0	8.5	0.0	3.0e-6	0.0	3.0e-6
eil51_n16_m15_uncorr-similar-weights_10	3948.211	245.4	0.0	0.0	8.7	5.8	1.5e-6	13.6	0.0
eil51_n17_m16_multiple-strongly-corr_01	685.565	248.6	0.0	0.0	8.4	0.2	1.5e-6	0.0	1.5e-6
eil51_n17_m16_multiple-strongly-corr_10	3826.098	2190.4	0.0	0.0	9.8	0.0	1.5e-6	0.0	1.5e-6
eil51_n17_m16_uncorr_01	2342.664	134.9	0.0	0.0	8.3	0.0	0.0	0.0	0.0
eil51_n17_m16_uncorr_10	2275.279	554.5	0.0	0.0	9.6	0.0	0.0	0.0	0.0
eil51_n17_m16_uncorr-similar-weights_01	556.851	70.8	0.0	0.0	8.1	0.0	0.0	0.0	0.0
eil51_n17_m16_uncorr-similar-weights_10	2935.961	787.7	0.0	0.0	9.7	0.0	0.0	0.0	0.0
eil51_n18_m17_multiple-strongly-corr_01	834.031	715.7	7.9	0.8	10.2	9.2	0.0	12.9	1.7e-6
eil51_n18_m17_multiple-strongly-corr_10	5531.373	6252.4	0.0	0.0	10.5	0.4	1.5e-6	0.4	1.5e-6
eil51_n18_m17_uncorr_01	2644.491	366.1	0.0	0.0	9.7	0.2	0.0	1.8	0.0
eil51_n18_m17_uncorr_10	3222.603	1462.7	0.0	0.0	10.3	0.0	1.3e-6	0.2	0.0
eil51_n18_m17_uncorr-similar-weights_01	532.906	148.3	0.0	0.0	8.5	0.0	1.3e-6	0.0	1.3e-6
eil51_n18_m17_uncorr-similar-weights_10	4420.438	1929.3	0.0	0.0	9.9	0.0	2.9e-6	0.3	1.8e-6
eil51_n19_m18_multiple-strongly-corr_01	910.229	1771.6	0.0	0.0	9.3	20.1	1.6e-6	20.1	1.6e-6
eil51_n19_m18_multiple-strongly-corr_10	-	-	-	-	10.4	-	-	-	-
eil51_n19_m18_uncorr_01	2604.844	830.2	0.0	0.0	9.7	0.0	0.0	0.0	0.0
eil51_n19_m18_uncorr_10	4048.408	3884.3	0.0	0.0	10.9	0.0	1.4e-6	0.0	1.4e-6
eil51_n19_m18_uncorr-similar-weights_01	472.186	412.3	0.0	0.0	9.2	0.0	1.5e-6	0.0	1.5e-6
eil51_n19_m18_uncorr-similar-weights_10	5573.695	5878.8	0.0	0.0	10.5	0.0	0.0	0.0	0.0
eil51_n20_m19_multiple-strongly-corr_01	518.189	4533.7	0.6	0.6	11.1	14.1	1.4e-6	12.3	0.0
eil51_n20_m19_multiple-strongly-corr_10	-	-	-	-	12.1	-	-	-	-
eil51_n20_m19_uncorr_01	2092.673	2456.9	0.0	0.0	8.7	0.0	0.0	0.0	0.0
eil51_n20_m19_uncorr_10	3044.391	12776.0	0.0	0.0	9.8	0.0	0.0	0.0	0.0
eil51_n20_m19_uncorr-similar-weights_01	451.052	1007.7	0.0	0.0	7.9	0.0	0.0	0.0	0.0
eil51_n20_m19_uncorr-similar-weights_10	4169.799	15075.7	0.0	0.0	9.4	0.0	0.0	0.0	0.0

Conclusions

- TTP is a multi-component problem combining TSP and KP.
- Many heuristic algorithms have been developed for TTP.
- We have shown exact approaches for PWT and TTP based on dynamic programming.
- Design gives insights into the interaction of the subproblems in TTP.
- Approaches allow to evaluate the quality achieved by state of the art heuristics.