TSP Neighbourhood reduction with POPMUSIC

Éric D. Taillard

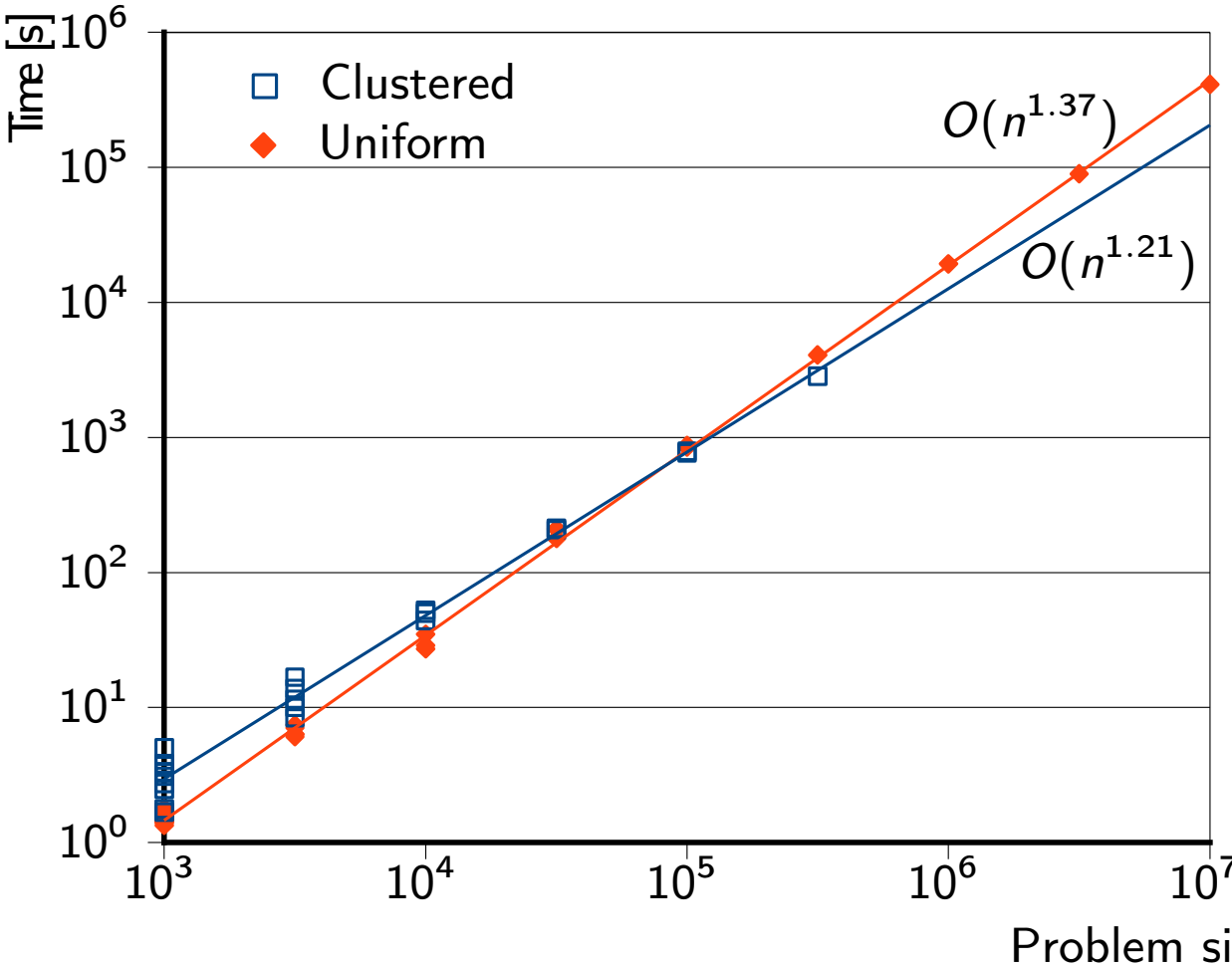
Barcelona, 6. July 2017

Tavelling Salesman Problem : State of the Art

- Exact algorithm
  - Concorde TSP solver (Applegate, Bixby, Chvátal, Cook, 2003)
  - Optimum known for 2D Euclidean instances with $\approx$ 100k cities
- Approximation algorithm
  - Local search with Lin & Kernighan neighbourhood (1973)
    + k-opt moves ($k = 5 \implies O(n^{4..})$ algorithmic complexity)
    + neighbourhood reduction (keep ≈ 5 edges for each vertex)
    + iterated with kick moves (double bridge)
  - One of the best heuristic implementation due to K. Helsgaun (LKH, 2009, 2014, 2016)
  - Solutions below 0.0...% over optimum for all problem of the literature (TSPLIB format, up to 10^7 cities

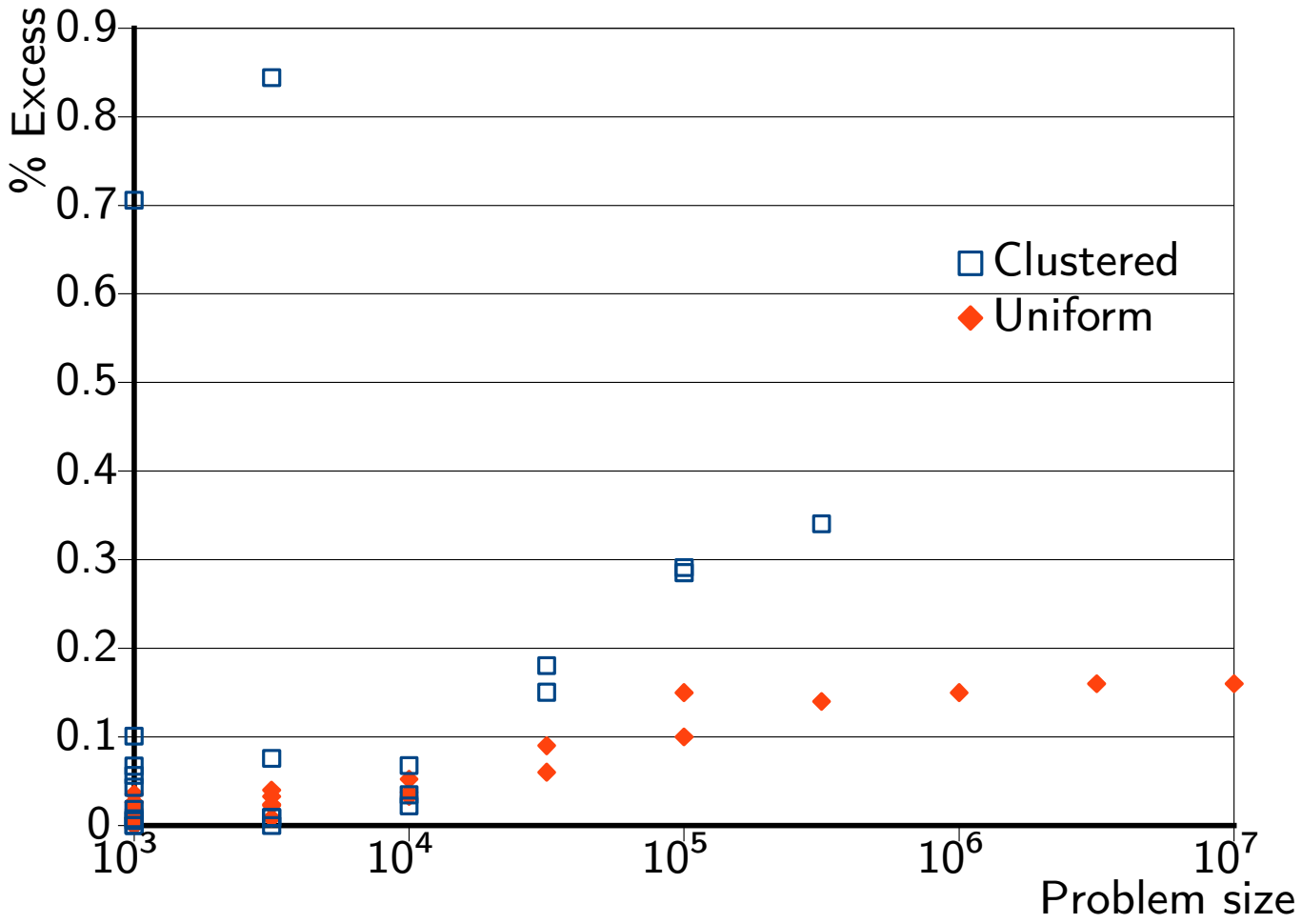
LKH (version 2.0.7) Speed, exploiting Euclidean property

Computational time of LKH. Uniform (E) and clustered (C) DIMACS instances. Delaunay triangulation + 4 quadrant-nearest neighbours as candidate edge list. 60 trials, single run.



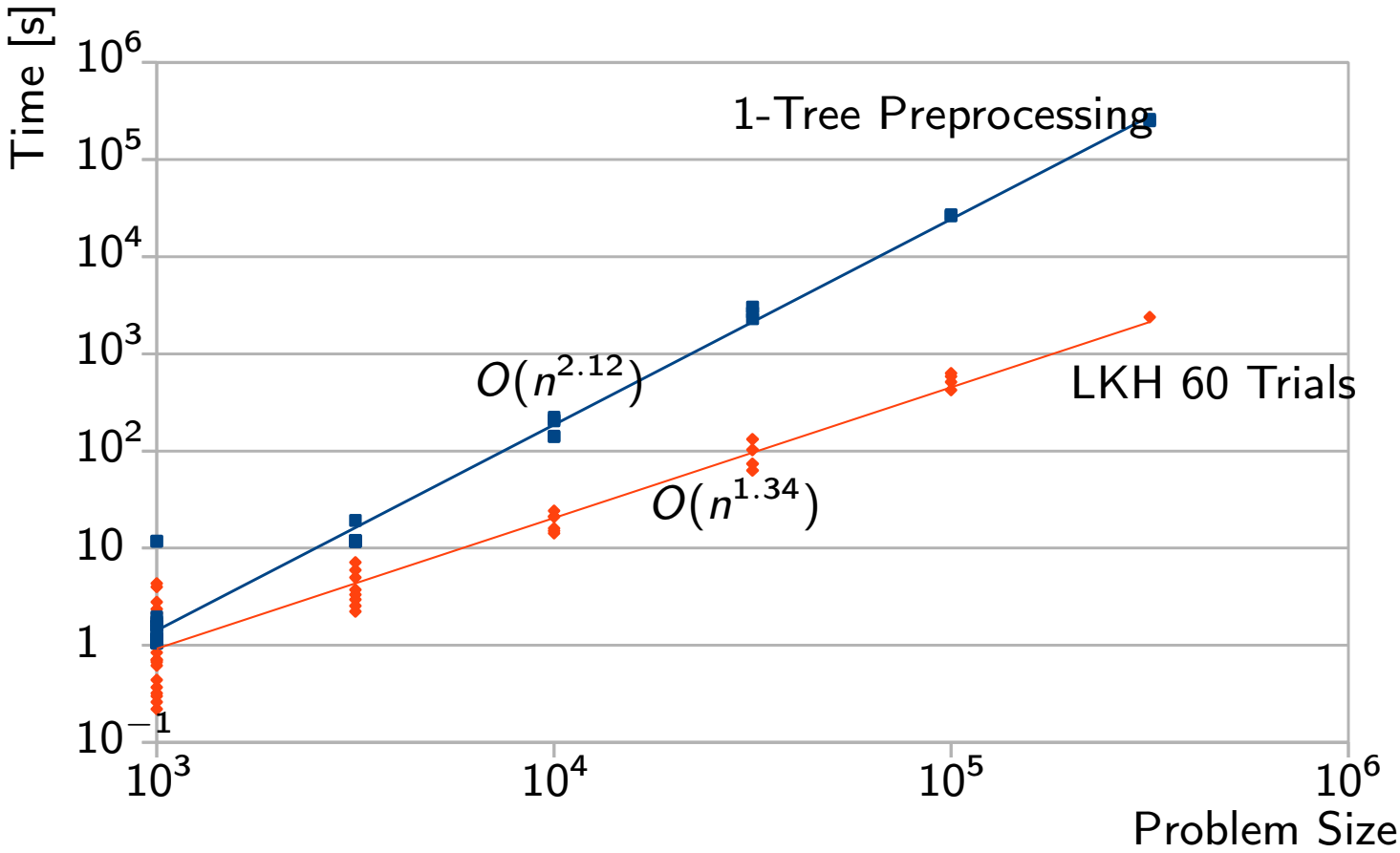
LKH Quality

Quality of solutions produces by LKH on uniform (E) and clustered (C) DIMACS instances. 60 Trials, 1 Run



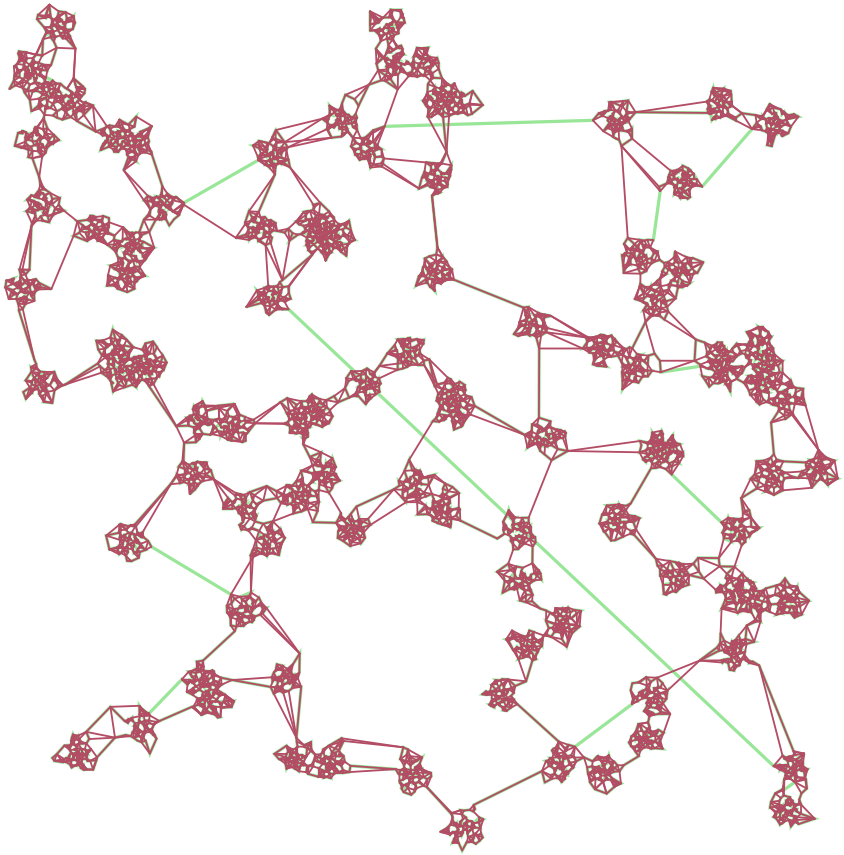
LKH General instances

- Standard preprocessing: 1-tree computation, Subgradient optimization
- Uniform (E) and clustered (C) DIMACS instances
- 60 trials, a single run



Potential problem with standard (1-tree) LKH preprocessing

- 5-opt moves are evaluated $\Longrightarrow$ $O(n^{4..})$ algorithmic complexity
- Preprocessing is absolutely necessary for limiting neighbourhood size



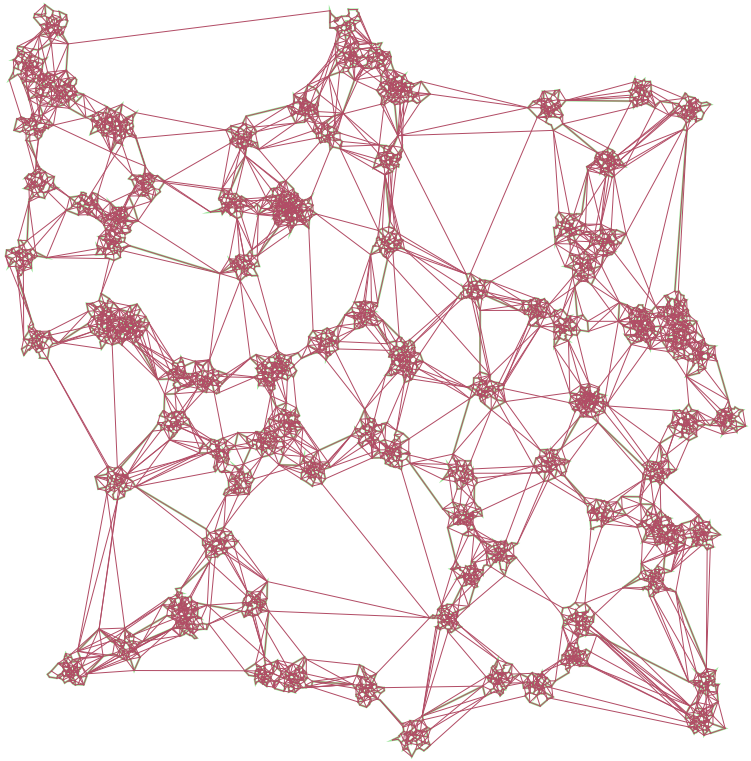
Tour merging

Generate several different tours with random components, keep only the edges of these tours

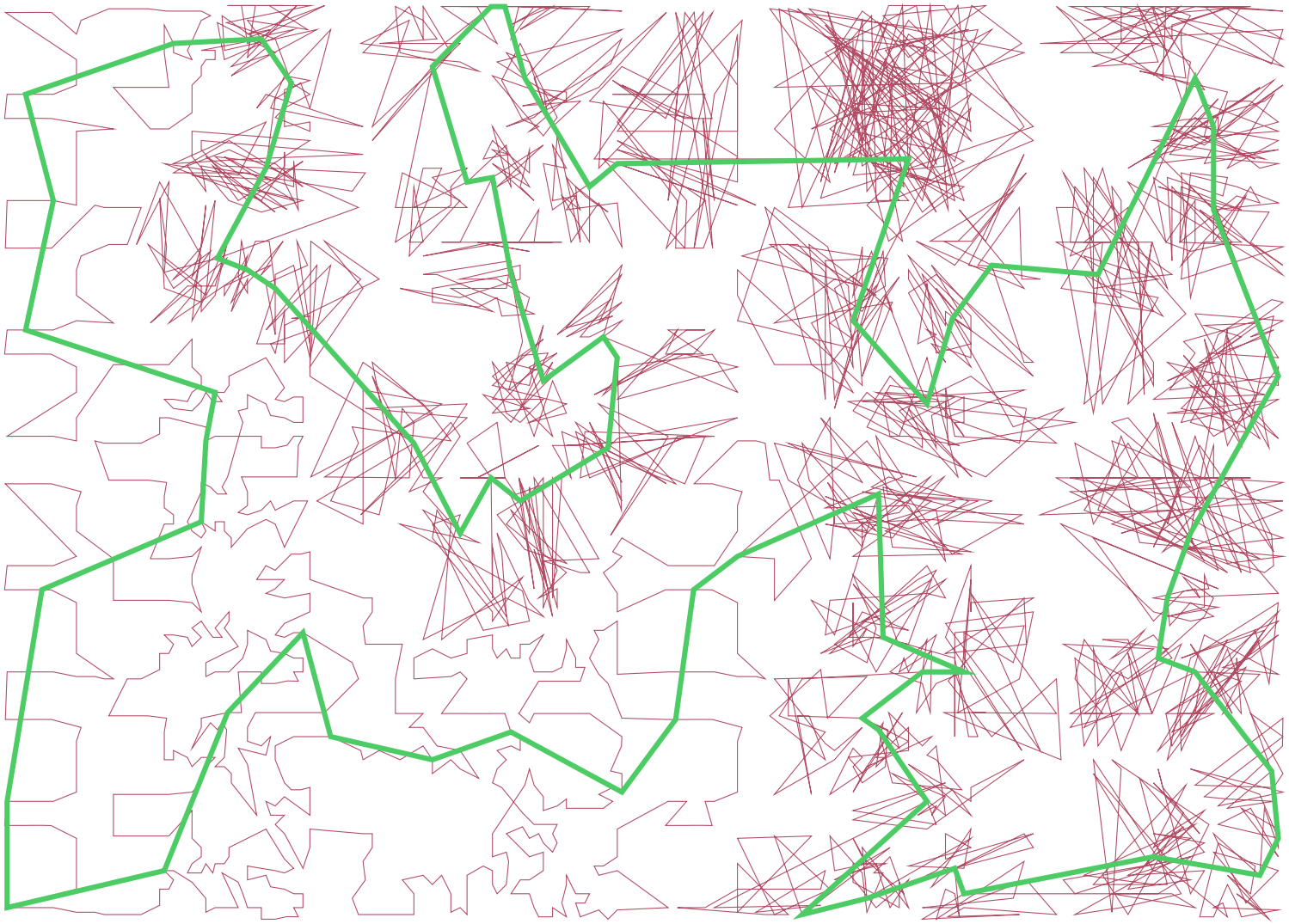
- Known to be efficient (implemented e.g. in Concorde)
- Improvement by Blazinskas and Misevicius (2012)
  - Start with a random tour
  - 3-opt neighbourhood with 40 nearest neighbour (10 in each quadrant)
  - Repeat 60 to 230 times (automatic, adaptive)
- Efficient ($< O(n^2)$) implementations only for Euclidean instances

Goal of the presentation

- Propose a general method extracting interesting edges
- The computational complexity should be lower than $O(n^2)$



Efficient solution building

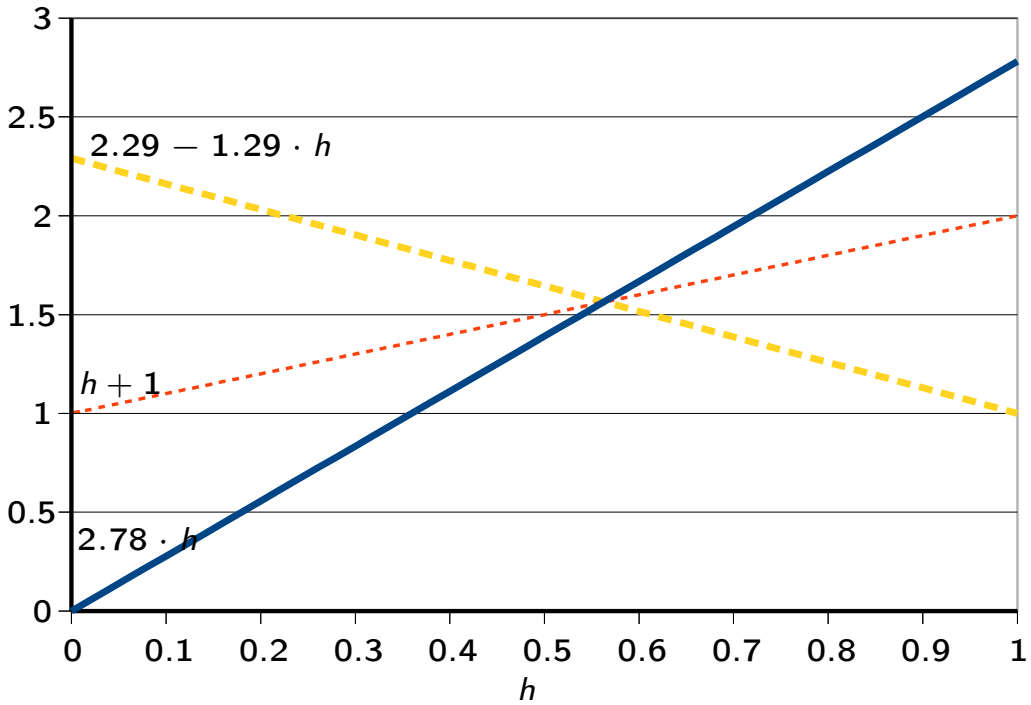


Obtaining a feasible solution for large instances

1: Select a sample S of a cities, randomly, uniformly, among the n cities; Build a LK-optimal tour T_s on S;
$T = T_s$
2: **for** each city $c \notin S$ (in arbitrary order) **do**
3: $c_s = argmin(d(s, c)), (s \in S)$;
4: Insert c just after c_s in T;
5: **end for**
6: **for** each city $c \in T_s$ **do**
7: Let n_c be the number of cities inserted at previous step after c and after the city next to c in T_s;
8: Optimize in T a sub-path of n_c cities starting from c with a 2-opt local search
9: **end for**

Empirical complexity of the algorithm

- Basic implementation of LK: $O(a^{2.78})$
- Basic implementation of 2-opt: $O((\frac{2n}{a})^{2.29})$
- $a \sim n^h$

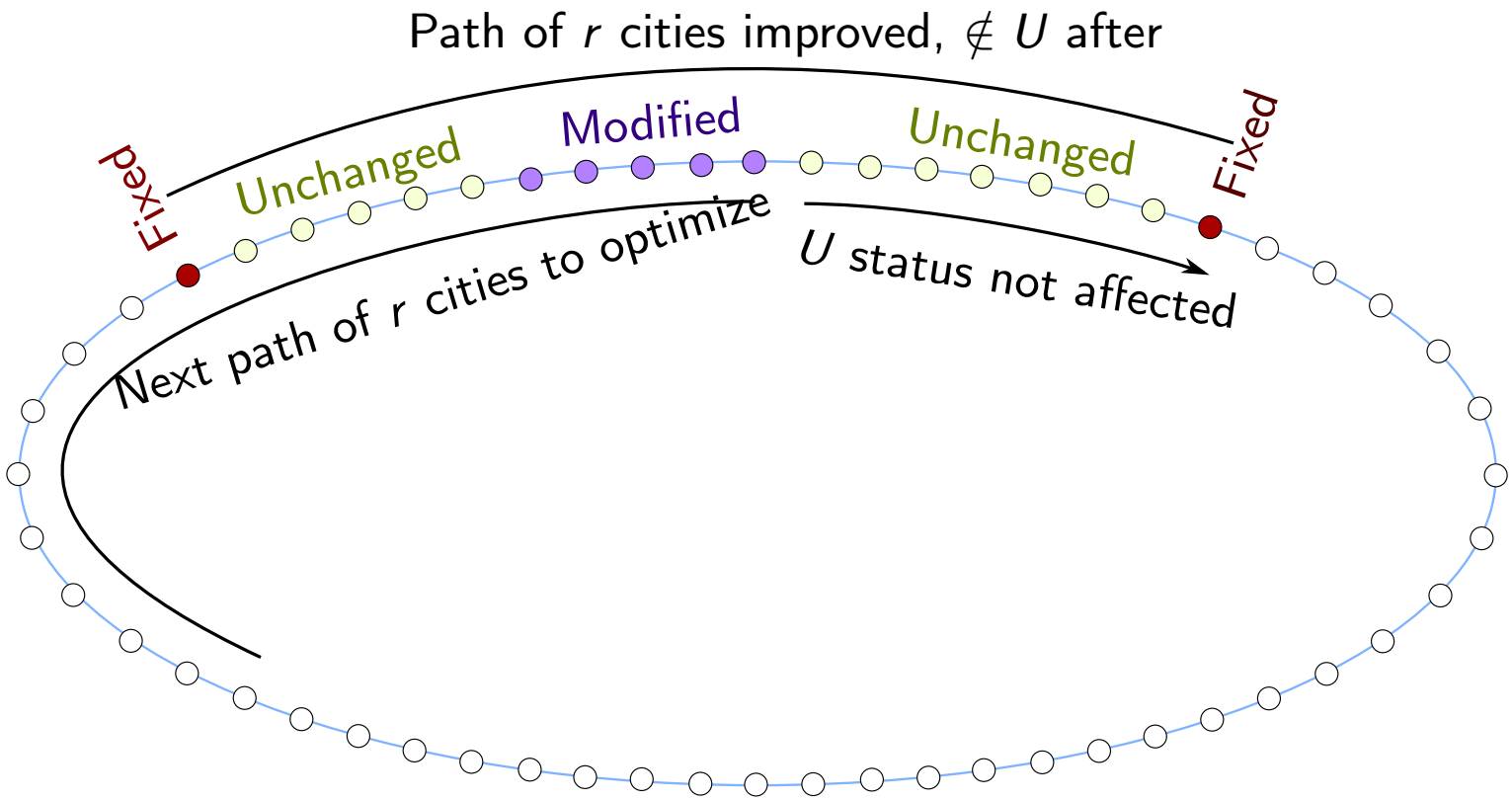


With $a = 1.5n^{0.56}$, the empirical complexity is in $O(n^{1.57})$

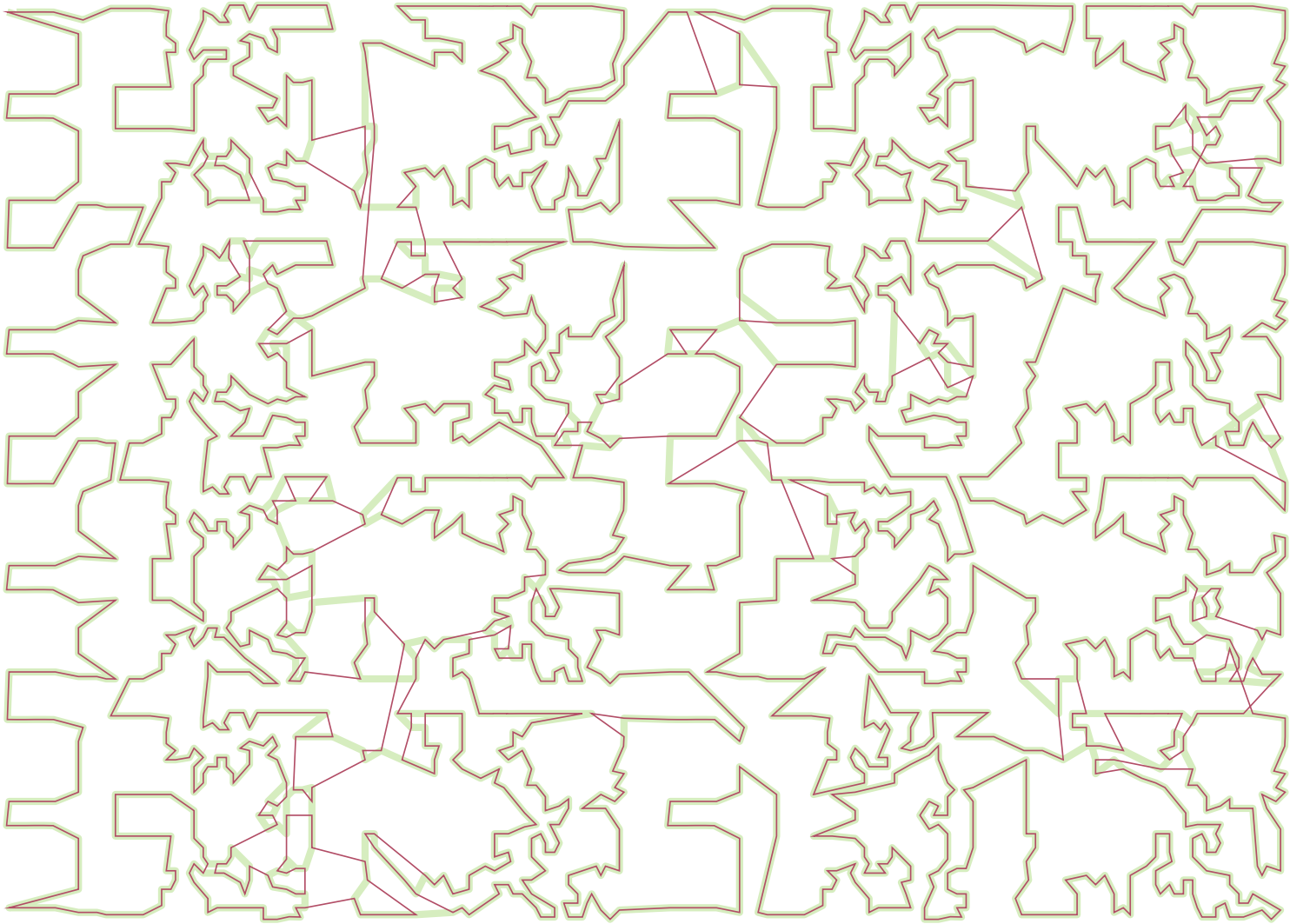
POPMUSIC improvement of a TSP solution

- Optimize every sub-paths of r cities with LK local search
- Put city i in a stack U, if a sub-path of r cities starting from i can potentially be improved with LK
- Stop when the stack is empty
- With $r = 50$, solutions so obtained are 5.7% to 12.8% above optimum on DIMACS (E and C) instances

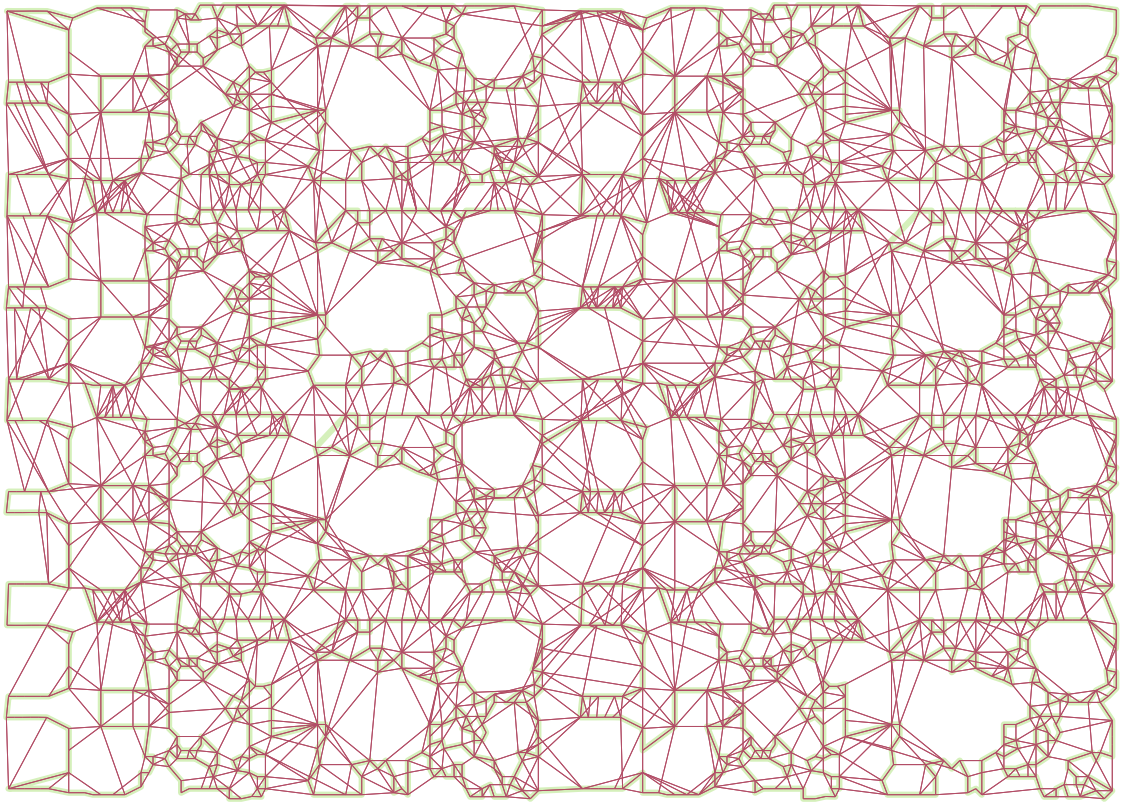
Identifying sub-path to improve



POPMUSIC solution (dark) and optimum solution (light)



Method for TSP neighbourhood reduction



- Get 20 different solutions with fast algorithm + POPMUSIC (random samples S)
- Union of these 20 solutions are high quality candidate edges for LKH

Computational results for DIMACS uniform (E) instances

Size	Best known	Computational Time (i7 930)			%Excess	
		Sub-graph	Total	LKH	Proposed	LKH
10^3	23101545.4	2.981	3.921	1.701	0.036	0.016
$10^{3.5}$	40519926	9.406	17.54	16.4	0.062	0.022
10^4	71865826	29.7	92.2	226	0.11	0.05
10^4	72031630	29.9	82.3	225	0.08	0.03
10^4	71822483	29.6	87.0	237	0.12	0.03
$10^{4.5}$	127282138	95.3	427	2396	0.19	0.06
$10^{4.5}$	126647285	95.2	470	2427	0.83	0.73
10^5	224330692	325	1850	26860	0.88	0.76
10^5	225654639	326	1805	27446	0.24	0.11
$10^{5.5}$	401301206	1234	7821	262090	0.26	0.15
10^6	713187688	5250	31061	—	0.27	—
$10^{6.5}$	1267318198	27770	129866	—	0.28	—
10^7	2253088000	208195	611402	—	0.28	—

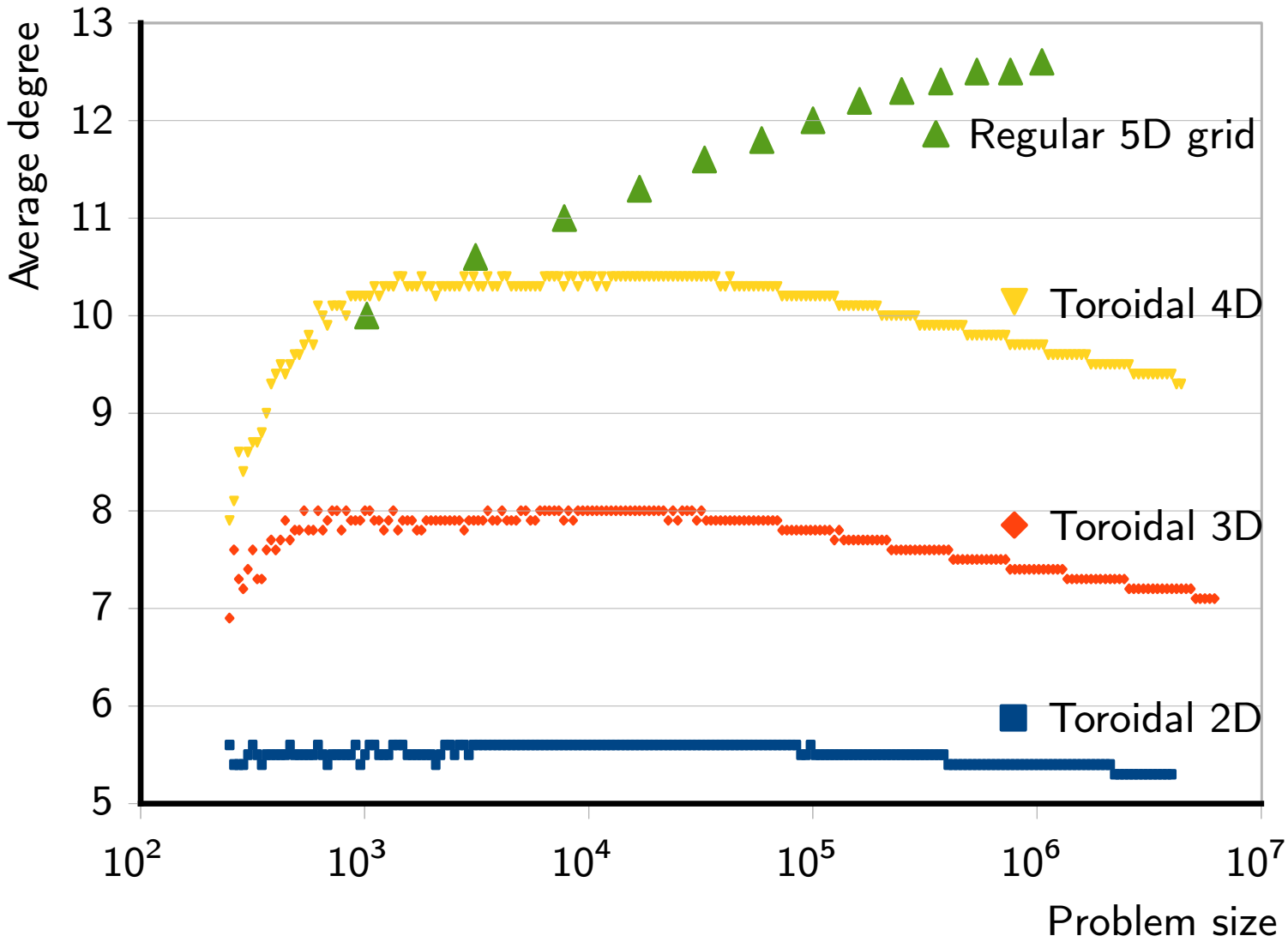
Computational results for DIMACS clustered (C) instances

Size	Best known	Computational Time (i7 930)			%Excess	
		Sub-graph	Total	LKH	Proposed	LKH
10^3	11174460.5	3.24	4.18	4.13	0.11	0.10
$10^{3.5}$	19147233.6	10.5	20.3	15.1	0.68	1.50
10^4	33001034	33.1	85.4	68	0.72	3.31
10^4	33186248	33.4	76.8	73.6	0.86	1.42
10^4	33155424	33.2	71.1	73.9	0.25	0.42
$10^{4.5}$	59545390	108	230	469	0.49	3.57
$10^{4.5}$	59293266	107	264	455	0.79	2.29
10^5	104617752	368	975	3457	1.16	4.56
10^5	105390777	368	947	3467	0.85	7.47
$10^{5.5}$	186870839	1372	3495	32018	0.98	9.44

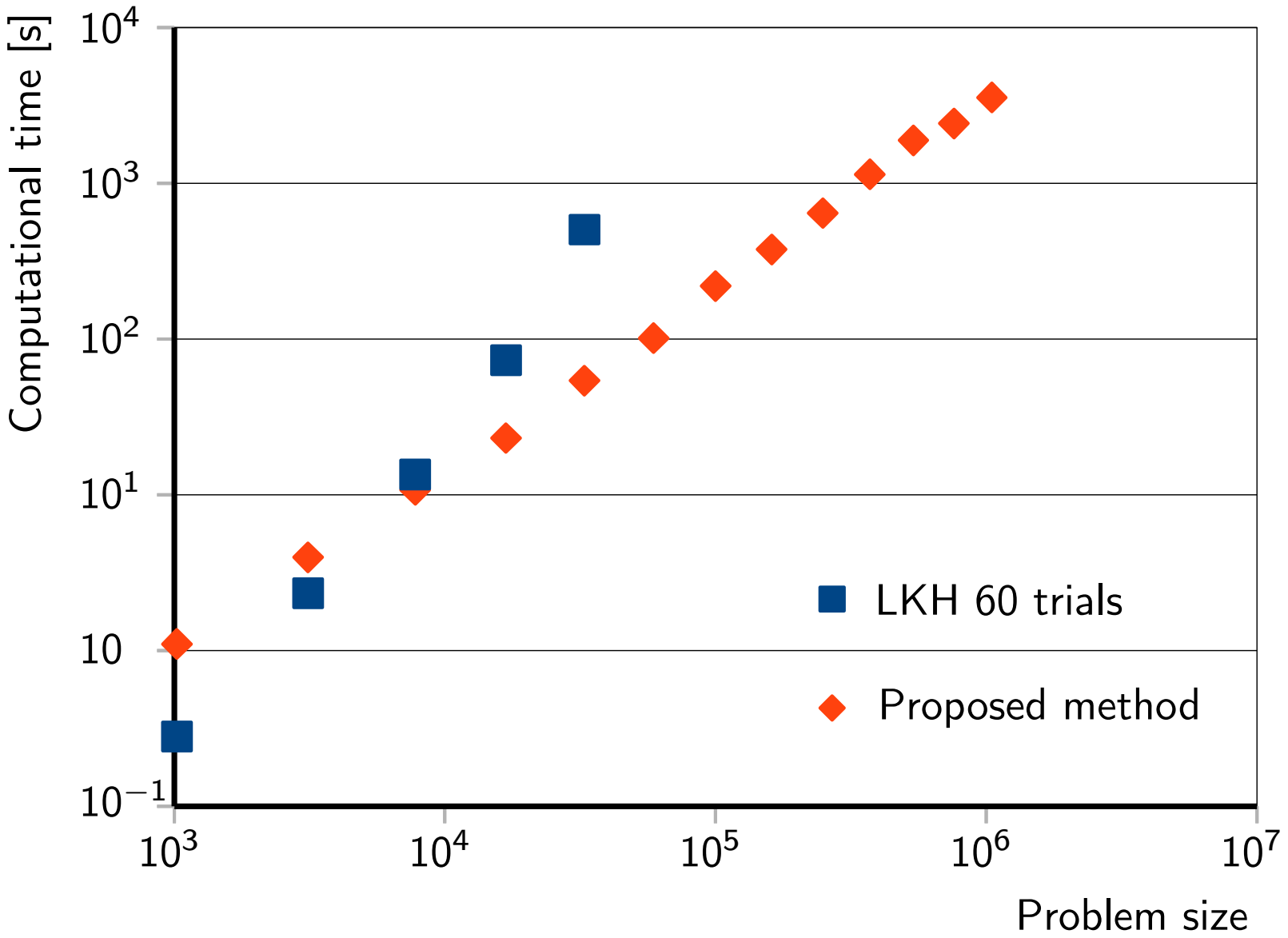
Conclusions and future work

- Recursive implementation of TSP solution building in $O(n\ \log(n))$
- Faster sub-path optimization procedure in POPMUSIC (3-opt with limited neighbourhood)
- Promising preliminary experiments
  - 10^9 cities TSP
  - Solution at less than 13% above predicted optimum
  - Less than 80 minutes on a standard PC (1 core)
  - Nearest neighbour would take 150 years on the same machine for getting a solution more than 20% above optimum
- Method competitive on 2D Euclidean instances, even if not exploiting the problem structure
- Experiments on different types of instances
- Incorporating POPMUSIC in LKH is on progress

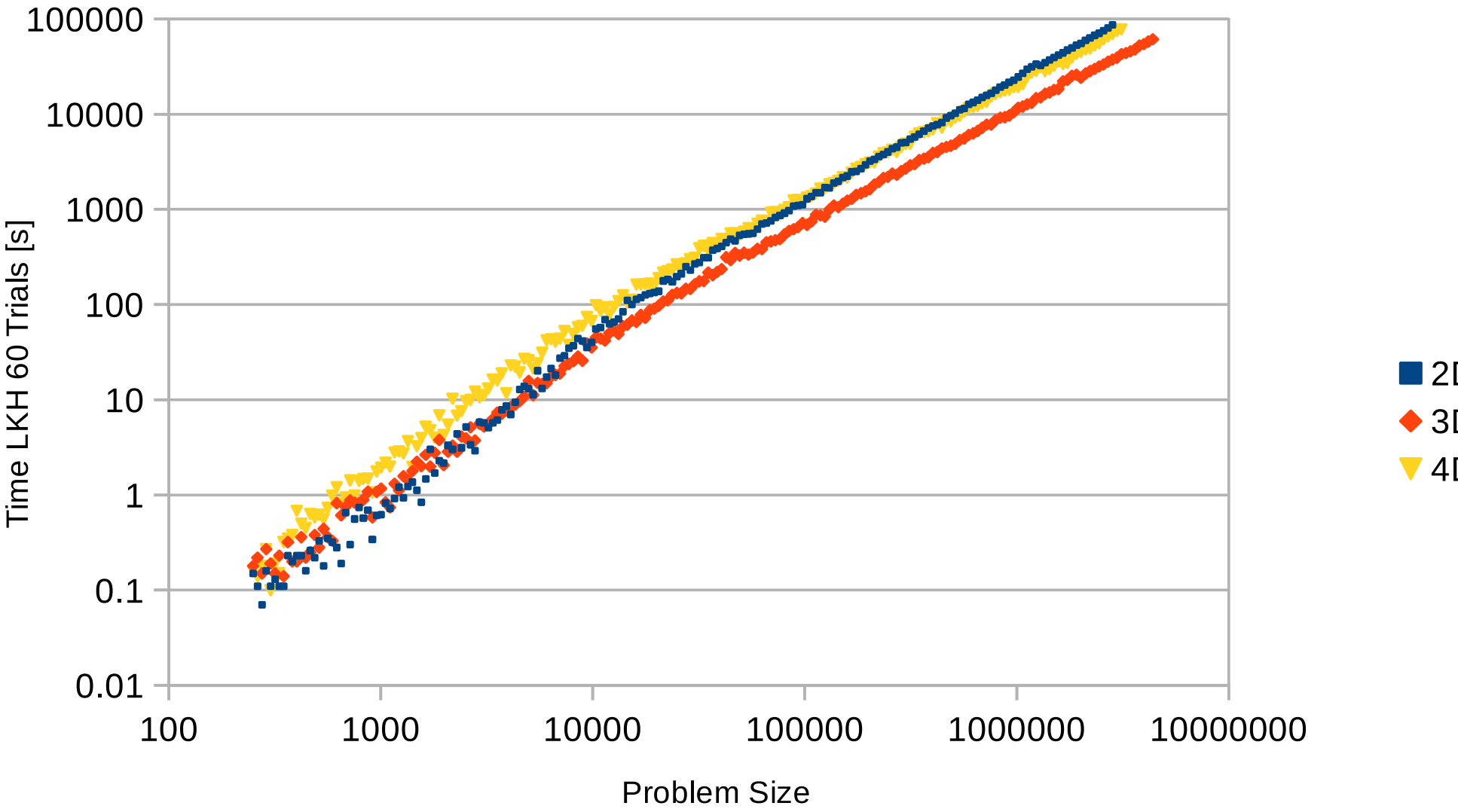
Average vertex degree, toroidal 2,3,4D, regular grid 5D



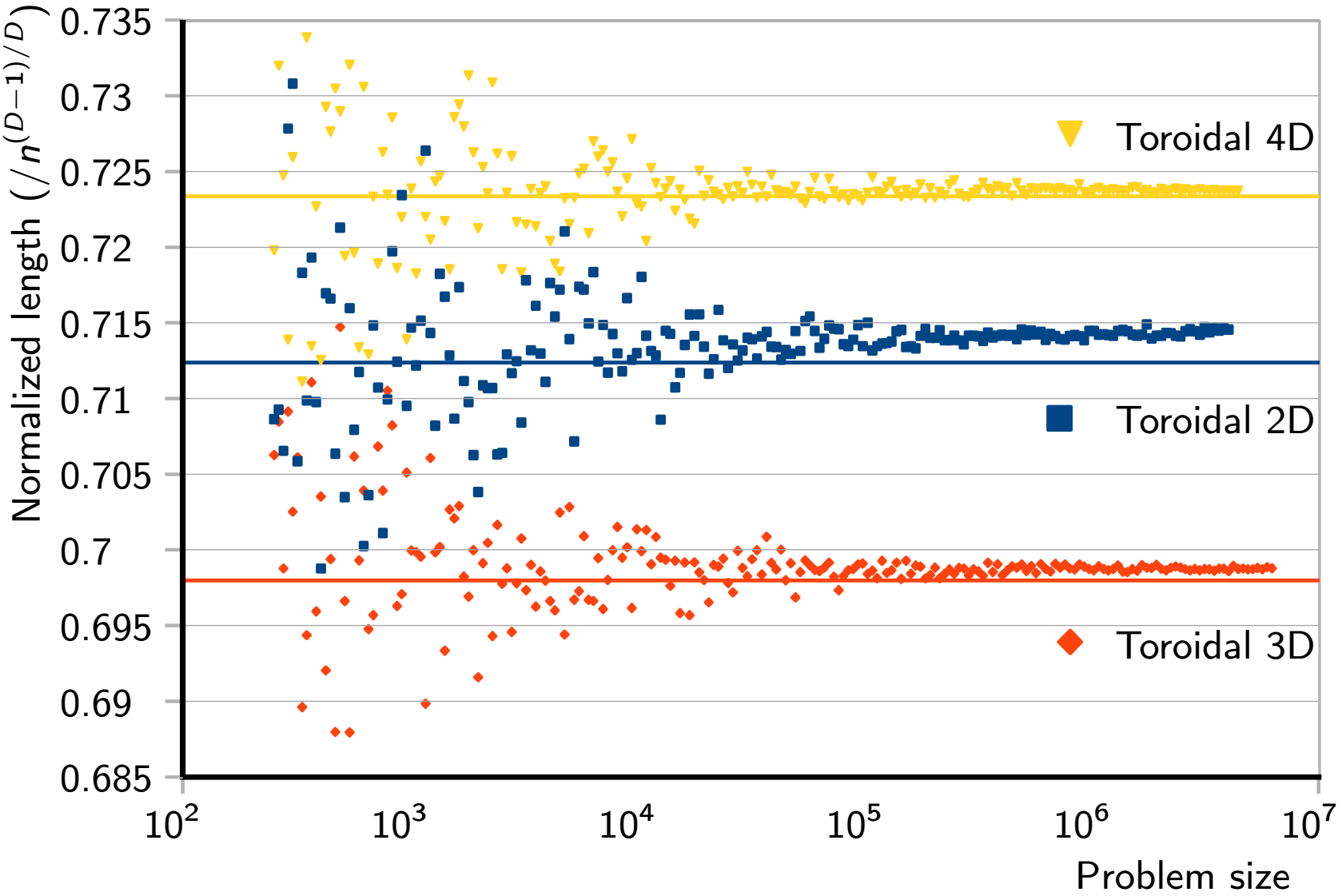
Computational time, Regular grid 5D



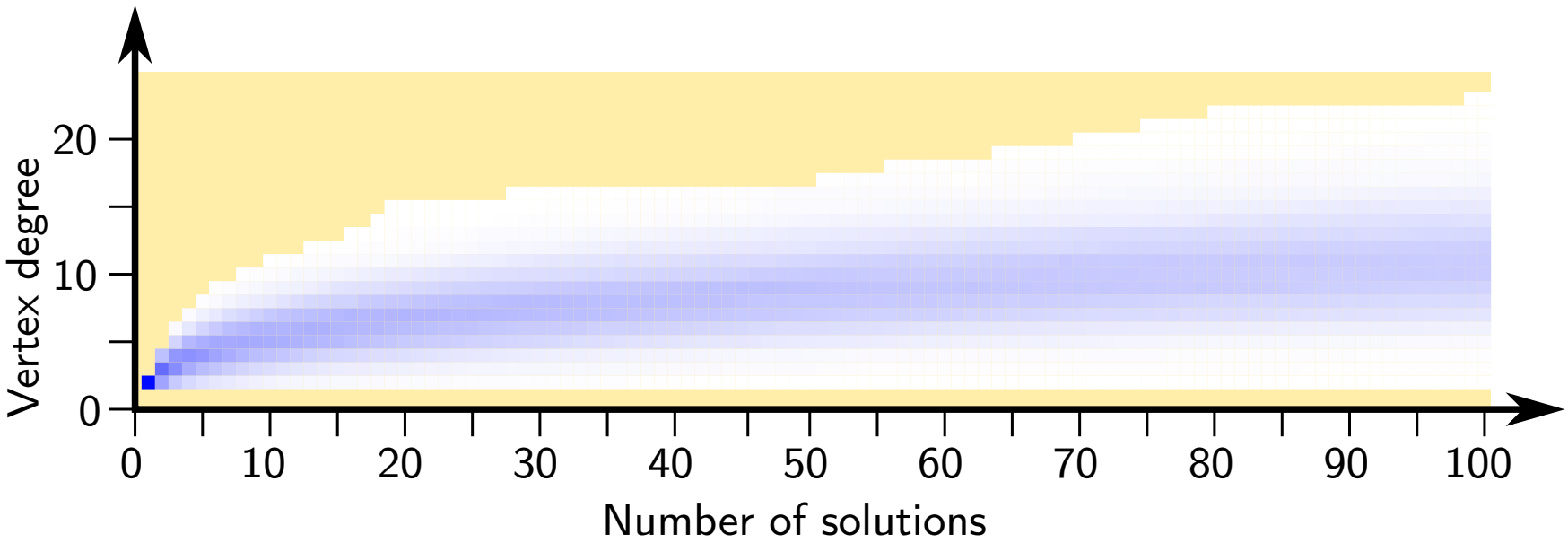
Computational time, Toroidal distances 2D, 3D, 4D



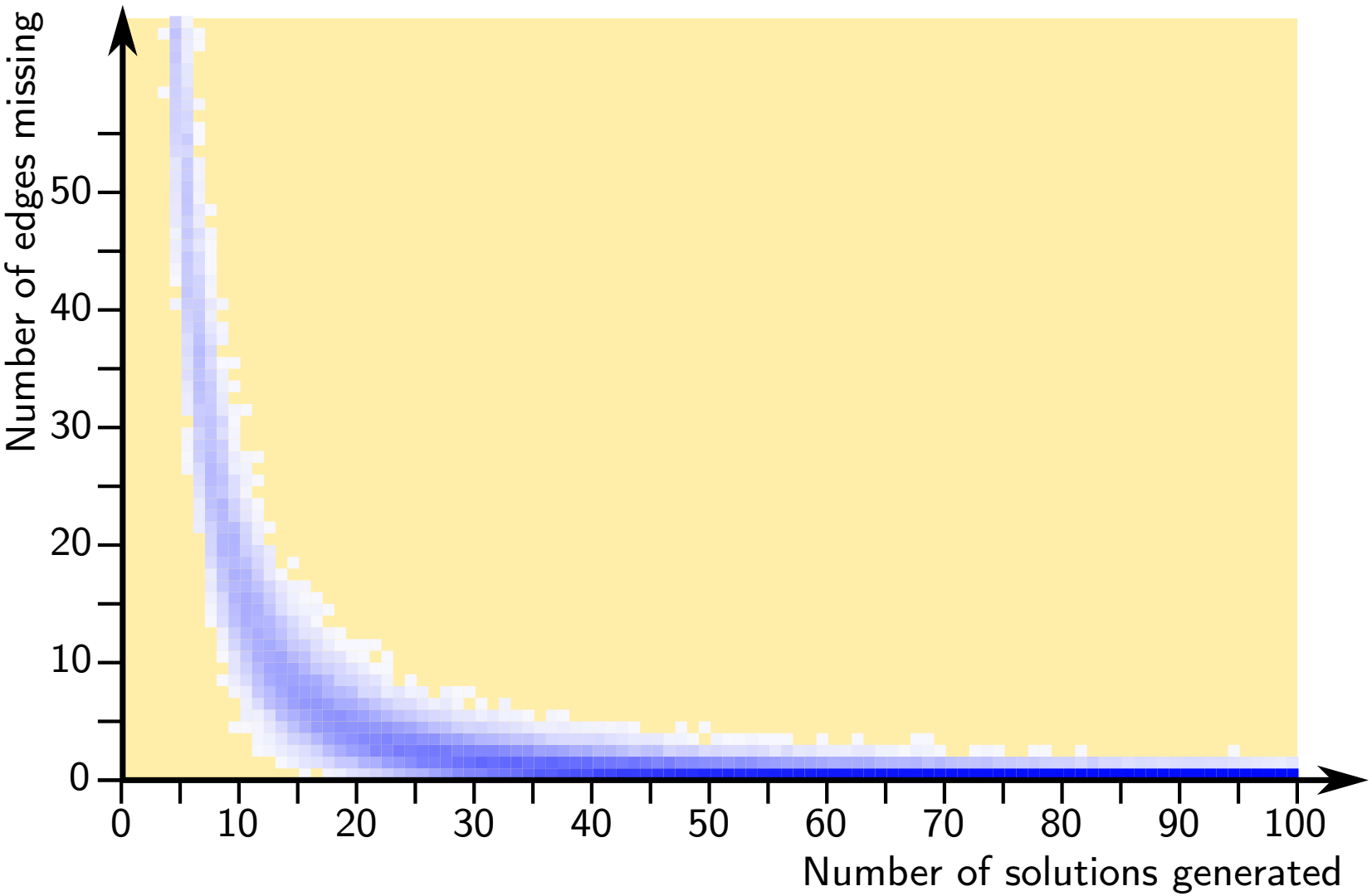
Solution quality Toroidal distances 2D, 3D, 4D



Vertex degree as a function of number of POPMUSIC solutions



Missing edges in sub-graph relative to optimum solution



Vertex degree for DIMACS instances, 20 POPMUSIC solutions