The Traveling Salesman Problem: State of the Art

Thomas Stützle

stuetzle@informatik.tu-darmstadt.de

http://www.intellektik.informatik.tu-darmstadt.de/˜tom.

Darmstadt University of Technology

Department of Computer Science

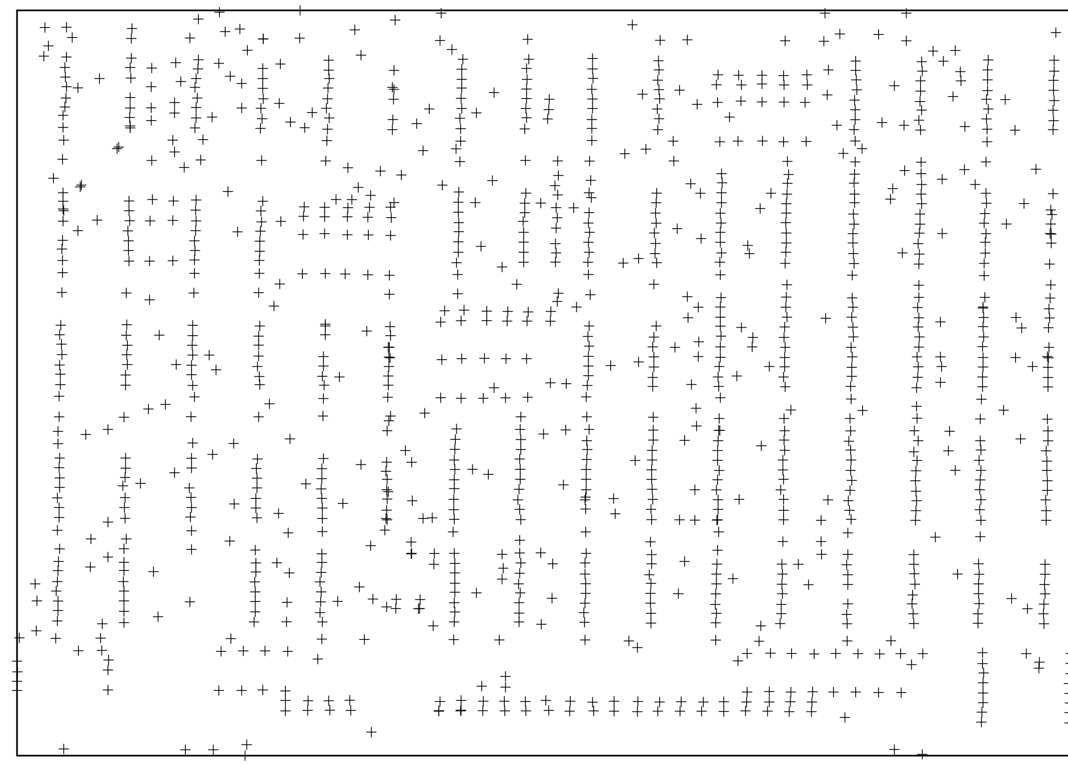
Intellectics Group

Outline

- TSP
- benchmarks
- complete algorithms
- construction heuristics
- local search
- SLS methods, ILS
- concluding remarks

Traveling Salesman Problem

- given: fully connected, weighted Graph $G = (V, E, d)$
- goal: find shortest Hamiltonian cycle
- hardness: $\mathcal{NP}$-hard
- interest: standard benchmark problem for algorithmic ideas



Why TSP?

The TSP is probably the most widely studied combinatorial optimization problem

- conceptually simple problem
- hard to solve ($\mathcal{NP}$-hard)
- didactic, design and analysis of algorithms not obscured by technicalities
- significant amount of research
- arises in a variety of applications

Benchmark instances

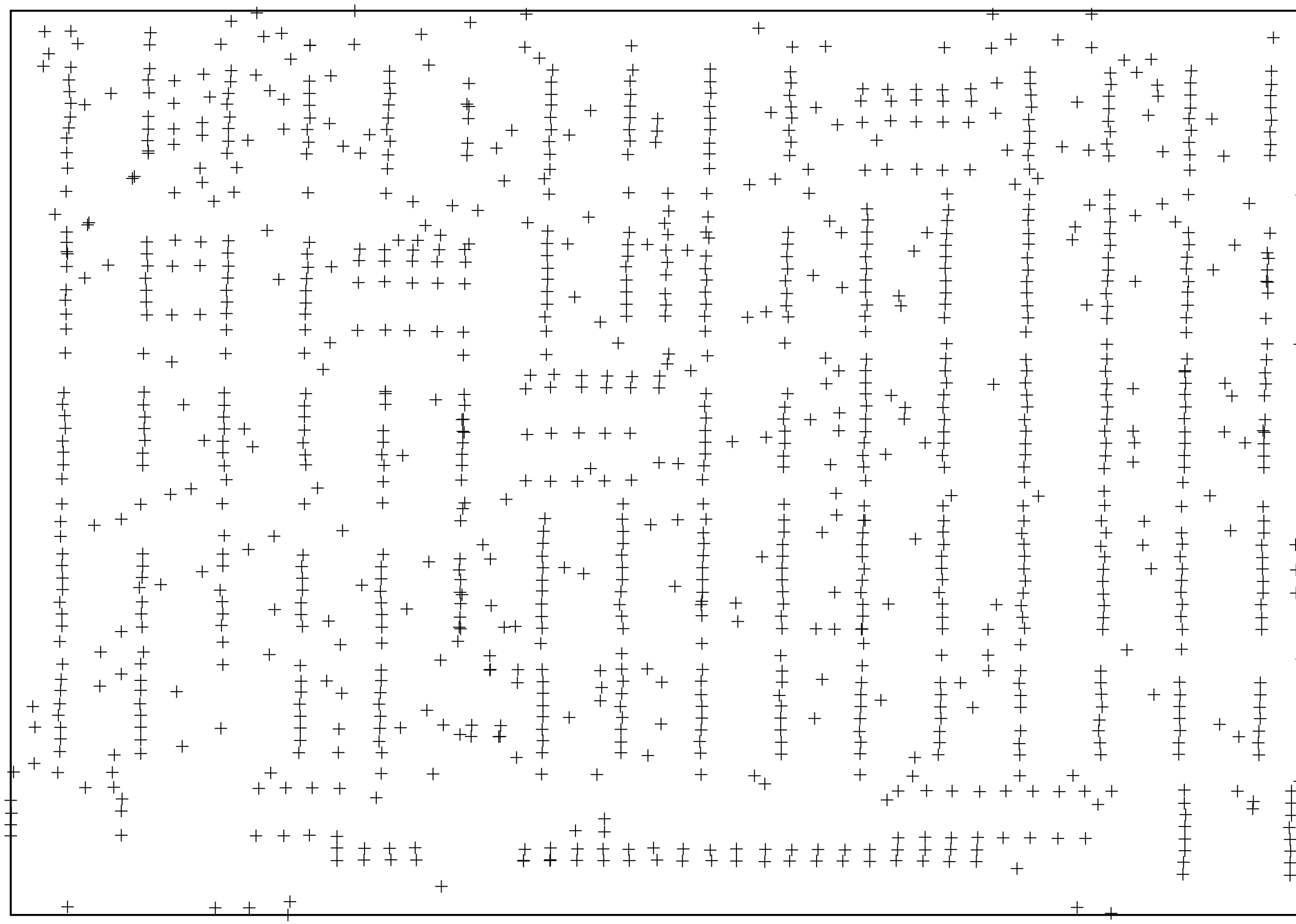
General considerations

- mainly metric TSPs, often Euclidean
- typically only integer distances

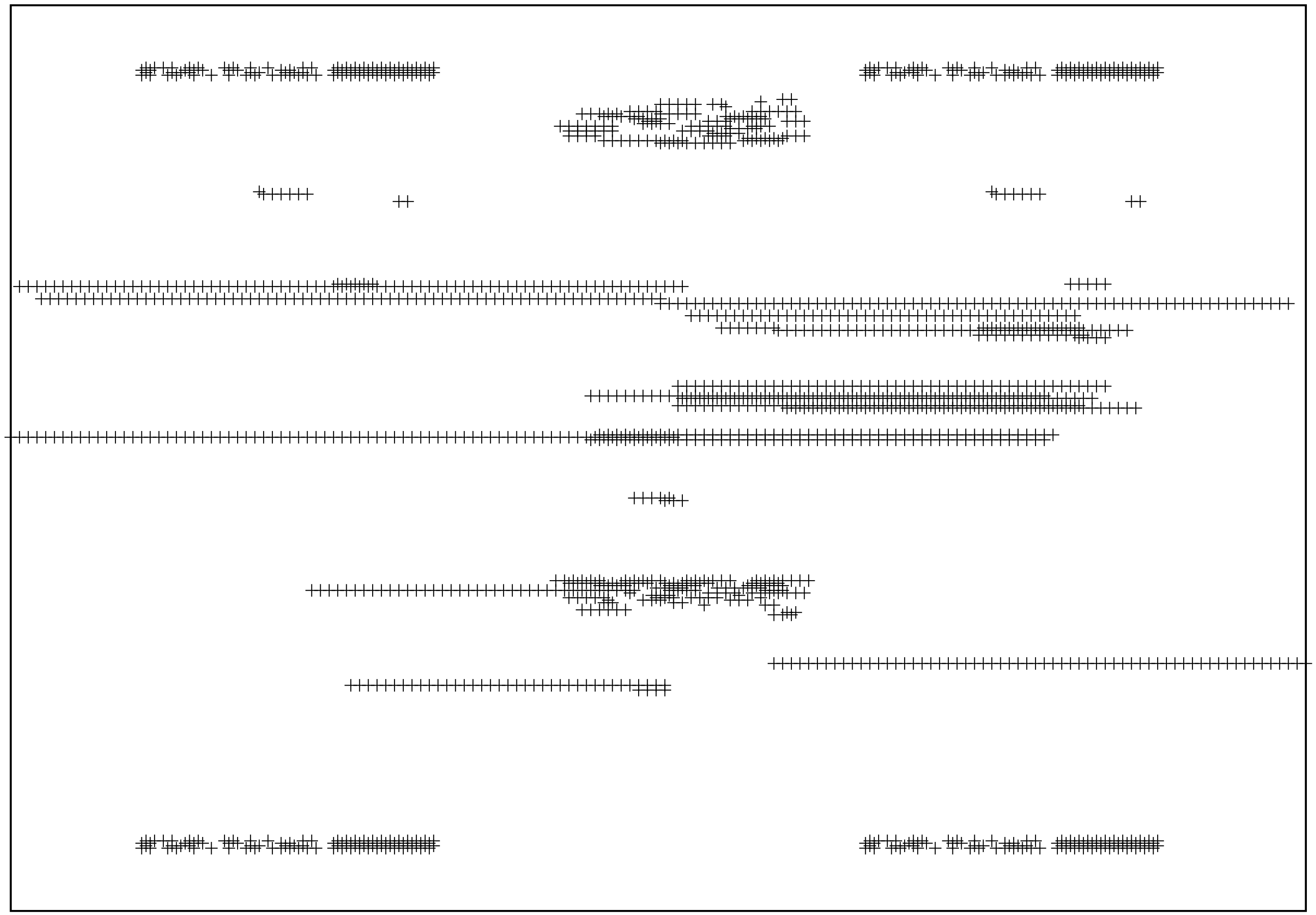
Instances

- TSPLIB
  - more than 100 instances upto 85.900 cities
  - some instances from practical applications
- instances from VLSI design
- random Euclidean instances (uniform and clustered)
  - some available from 8th DIMACS challenge

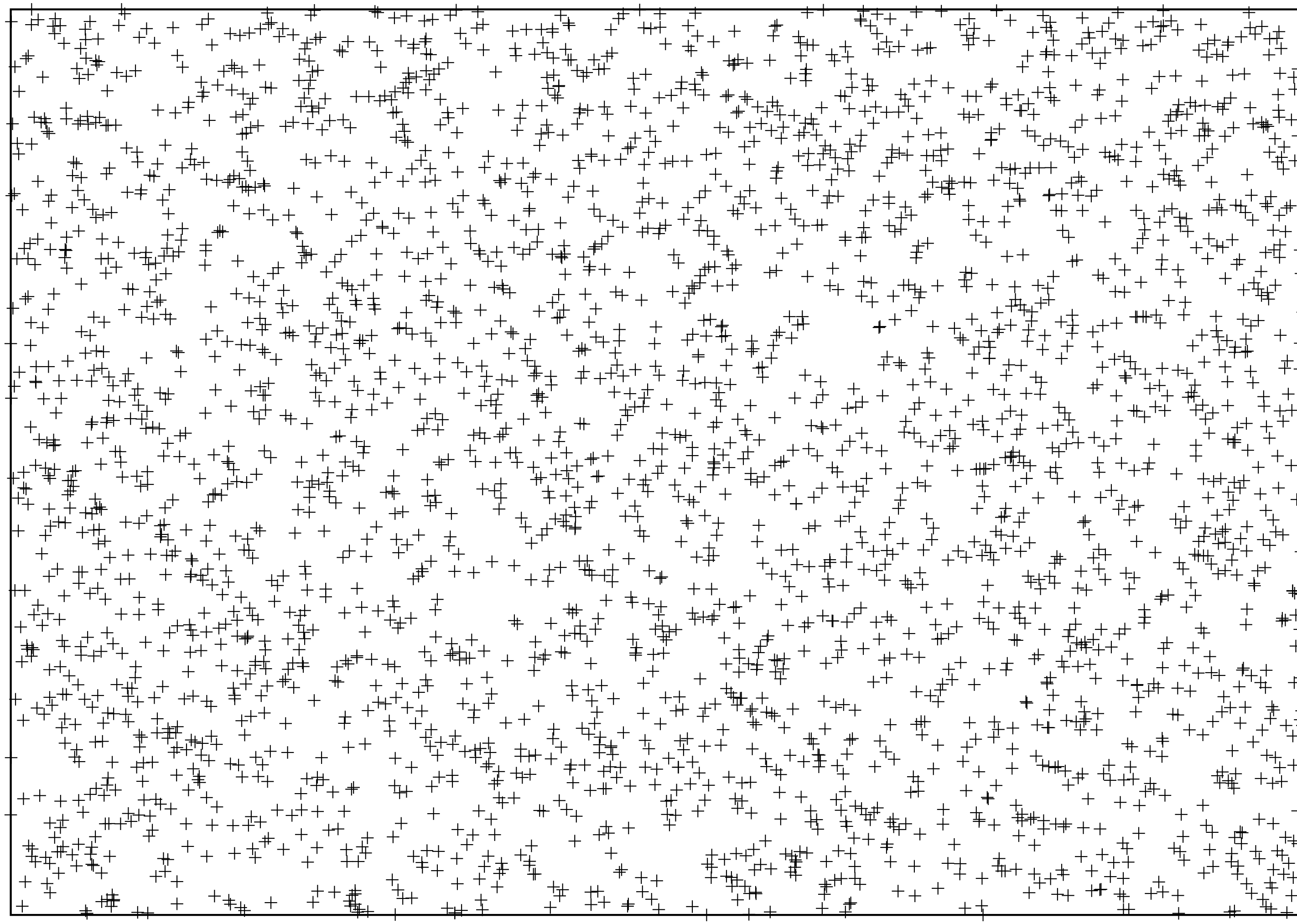
Benchmark instances



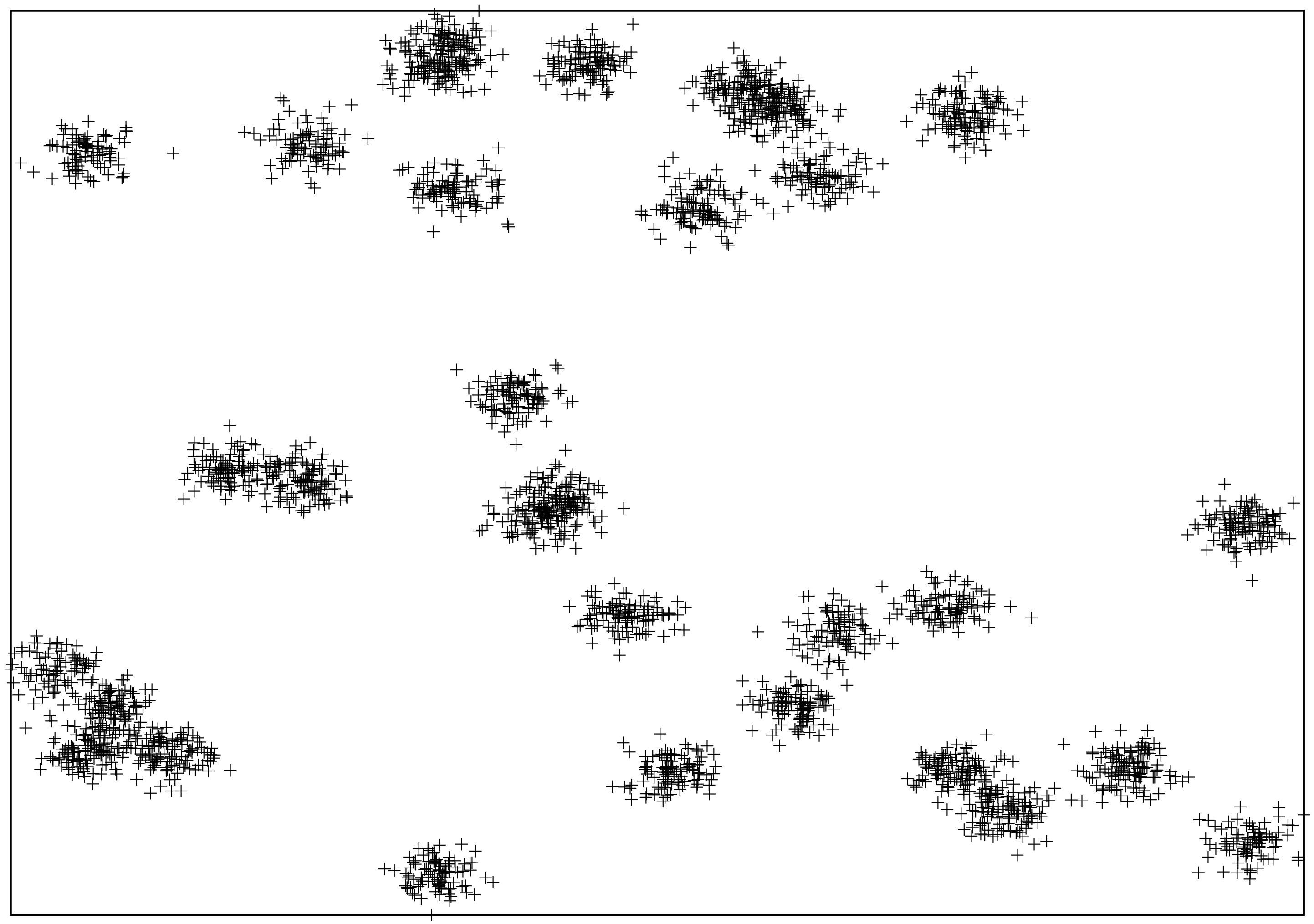
Benchmark instances



Benchmark instances



Benchmark instances



Lower bounds

- lower bounds are necessary to rate quality guarantees of tours
- are needed within complete algorithms
- best lower bound: Held-Karp bounds
  - experimentally, shown to be very tight
  - within less than one percent of optimum for random Euclidean
  - up to two percent for TSPLIB instances

Complete algorithms

- branch & bound algorithms
- branch & cut algorithms (state-of-the-art)
  - use LP-relaxation for lower bounding schemes
  - effective heuristics for upper bounds
  - branch if cuts cannot be found easily
- state-of-the-art
  - largest instance solved has 15.112 cities!
  - often instances with few thousands of cities can be regularily solved within minutes / hours
  - Concorde code publically available

Complete algorithms

Solution times with Concorde

Instance	No. nodes	CPU time (secs)
`att532`	7	109.52
`rat783`	1	37.88
`pcb1173`	19	468.27
`fl1577`	7	6705.04
`d2105`	169	11179253.91
`pr2392`	1	116.86
`rl5934`	205	588936.85
`usa13509`	9539	ca. 4 years
`d15112`	164569	ca. 22 years

Construction heuristics

- grow tours by iteratively adding cities to partial tours
- many variants available
  - nearest neighbor heuristics (ca. 22% from opt)
  - insertion heuristics (ca. 14% from opt; farthest insertion)
  - greedy heuristics (ca. 14% from opt)
  - savings heuristic (ca. 12 % from opt)
  - Christofides heuristic (ca. 10% from opt)
- good compromise between solution quality and computation time by greedy or savings heuristic

Local search algorithms

- k-exchange heuristics
  - 2-opt
  - 2.5-opt
  - 3-opt
- complex neighborhoods
  - Lin-Kernighan
  - Helsgaun’s Lin-Kernighan variant
  - Dynasearch
  - ejection chains approach
- all exploit TSP specific and general implementation and speed-up techniques

Implementation

- typically all algorithms use first-improvement
- neighborhood pruning
  - fixed radius nearest neighbor search
- neighbor lists
  - restrict exchanges to most interesting candidates
- don’t look bits
  - focus local search to "interesting" part
- sophisticated data structures
- extremely large instances tackable
  (largest had 25 million cities!)

Example results: TSP

- timings for 1000 local searches with `2-opt` and `3-opt` variants from random initial solutions on a Pentium III 500 MHz CPU. **std**: no speed-up techniques; **fr+cl**: fixed radius and unbounded candidate lists, **dlb**: don't look bits

	`2-opt`-**std**		`2-opt`-**fr+cl**		`2-opt`-**fr+cl+dlb**		`3-opt`-**fr+cl+dlb**	
instance	Δavg	t_{avg}	Δavg	t_{avg}	Δavg	t_{avg}	Δavg	t_{avg}
`kroA100`	8.9	1.6	6.4	0.5	6.6	0.4	2.4	4.3
`d198`	5.7	6.4	4.2	1.2	4.3	0.8	1.4	30.1
`lin318`	10.6	22.1	7.5	2.1	7.9	1.5	3.4	65.5
`pcb442`	12.7	55.7	7.1	2.9	7.6	2.2	3.8	63.4
`rat783`	13.0	239.7	7.5	7.5	8.0	5.8	4.2	213.8
`pr1002`	12.8	419.5	8.4	13.2	9.2	9.7	4.6	357.6
`pcb1173`	14.5	603.1	8.5	16.7	9.3	12.4	5.2	372.3
`d1291`	16.8	770.3	10.1	16.9	11.1	12.4	5.5	377.6
`fl1577`	13.6	1251.1	7.9	25.8	9.0	19.2	4.0	506.8
`pr2392`	15.0	2962.8	8.8	65.5	10.1	49.1	5.3	878.1

Lin-Kernighan heuristic

- complex moves are build as being a concatenation of a number of simple moves
- the number of simple moves composing a complex one is variable and determined based on gain criteria
- the simple moves need not be independent of each other
- termination is guaranteed through additional conditions on the simple moves

Example results: `3-opt` *and LK*

- taken from DIMACS Challenge results, normalized times on a Compaq DS20 500 Mhz Alpha EV6

	`3-opt`-JM		LK-JM		LK-ABCC		LK-H	
instance	Δ_{avg}	t_{avg}	Δ_{avg}	t_{avg}	Δ_{avg}	t_{avg}	Δ_{avg}	t_{avg}
`pcb1173`	3.09	0.12	0.87	0.25	2.14	0.10	0.18	6.69
`fl1577`	5.65	0.20	0.93	8.72	9.01	0.15	5.56	14.86
`pr2392`	3.34	0.27	1.85	0.77	3.30	0.18	0.34	34.87
`rl5915`	2.44	0.83	1.22	3.29	3.32	0.46	0.39	242.99
`d15112`	2.30	1.99	1.13	4.60	1.82	2.38	0.11	1515.99
`pla85900`	3.80	6.87	1.21	46.20	1.21	8.84	0.85	48173.84

Lin-Kernighan heuristic

- good Lin-Kernighan implementations are the best performing local searches for TSP
- many variants of the algorithm are available (see also recent DIMACS challenge)
- an efficient implementation requires sophisticated data structures (several articles available on this subject)
- implementation is quite time consuming, but:
  at least three very good implementations are publically available (Concorde, Helsgaun, Neto)

Hybrid SLS methods

- Hybrid SLS methods are required when very high solution is desired

available (good working) approaches

- iterated local search
- approaches using ILS as a subroutine
  - tour merging
  - multi-level algorithms
- memetic algorithms
- ant colony optimization

ILS Example — TSP

basic ILS algorithm for TSP

- GenerateInitialSolution: greedy heuristic
- LocalSearch: `2-opt`, `3-opt`, `LK`, (whatever available)
- Perturbation: double-bridge move (a 4-opt move)
- AcceptanceCriterion: accept $s^{*\prime}$ only if $f(s^{*\prime}) \leq f(s^*)$

ILS for TSPs

- iterated descent *Baum, 1986*
  - first approach, relatively poor results
- large step Markov chains *Martin, Otto, Felten, 1991, 1992, 1996*
  - first effective ILS algorithm for TSP
  - introduced double-bridge move for ILS
  - simulated annealing type acceptance criterion
- iterated Lin-Kernighan *Johnson, 1990, 1997*
  - efficient ILS implementation based on preprints of MOF91
  - efficient LK implementation
  - accepts only shorter tours
  - slightly different perturbation from MOF
- data perturbation *Codenotti et.al, 1996*
  - complex perturbation based on changing problem data
  - good LK implementation

ILS for TSPs

- improved LSMC *Hong, Kahng, Moon, 1997*
  - study of different perturbation sizes, acceptance criteria with `2-opt`, `3-opt`, `LK` local search
- CLO implementation in Concorde
  Applegate, Bixby, Chvatal, Cook, Rohe, 199?-today
  - very fast `LK` implementation, publicly available, applied to extremely large instances (25 million cities!)
  - various perturbations available
- ILS with fitness-distance based diversification
  Stützle, Hoos 1999
  - diversification mechansim in ILS for long run times
  - very good performance with only `3-opt` local search
- ILS with genetic transformation *Katayama, Narisha, 1999*
  - perturbation guided by a second solution

ILS for TSPs

- Iterated Helsgaun
  - extremely effective LK implementation based on 5-exchange moves
  - constructive mechanism for generating new starting tours (no double-bridge move)
  - new way of constructing very small candidate lists
  - found many new best tours for large instances
- Iterated LK variant by Nguyen et al.
  - effective LK implementation based on Helsgaun's ideas
  - much faster than Iterated Helsgaun
  - new way of constructing very small candidate lists

Memetic algorithms for TSPs

Just to mention a few ..

- Gorges-Schleuter, Mühlenbein (1989)
- Braun (1992?)
- Nagata, Kobayashi (edge-assembly crossover, 1997)
- Merz, Freisleben (DPX-crossover, greedy crossover, 1996–2002)
- Möbius et al. (Iterative partial transcription, 1999)
- Houdayer, Martin (recursive MA, 1999)

it is not clear whether these algorithms are able to beat good ILS algorithms when very effective local searches are used

Example results: ILS and extensions

- taken from DIMACS Challenge results, normalized times on a Compaq DS20 500 Mhz Alpha EV6

	ILS-JM		ILS-ABCC		ILS-H		TourMerging	
instance	Δavg	t_{avg}	Δavg	t_{avg}	Δavg	t_{avg}	Δavg	t_{avg}
`pcb1173`	0.14	7.09	0.28	1.45	0.18	23.6	0.0	30.8
`fl1577`	0.69	311.76	7.05	6.0	0.06	440.5	0.02	161.7
`pr2392`	0.16	19.2	0.52	3.5	0.0	150.9	0.0	92.5
`rl5915`	0.48	129.7	0.77	13.9	0.04	1118.2	0.02	704.5
`d15112`	0.24	332.1	0.20	74.6	0.02	26322.9	—	—
`pla85900`	0.27	6127.8	0.29	500.0	—	—-	—	—

State-of-the-art

General results for TSP algorithms

- complete algorithms can solve surprisingly large instances
- construction heuristics and iterative improvement can be applied to very large instances ($> 10^6$ cities) with considerable success
- best performance results w.r.t. solution quality obtained with iterated local search, genetic algorithms, or more TSP-specific approaches
- best performing hybrid algorithms use 3-opt or Lin-Kernighan local search
- 8th DIMACS Implementation Challenge on the TSP gives overview of state–of–the–art results

Conclusions

The TSP ...

- has been a source of inspiration for new algorithmic ideas
- is a standard test-bed for complete and incomplete algorithms

Recent contributions to TSP solving:

- pushing the frontier of tractable instance size
- finding candidate solutions of very high quality
- better understanding of algorithm behaviour

References

- Holger Hoos and Thomas Stützle. *Stochastic Local Search*, Chapter 8 (TSP), Morgan Kaufmann Publishers, 2004.
- D. S. Johnson and L. A. McGeoch. The travelling salesman problem: A case study in local optimization. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley & Sons, Chichester, UK, 1997.
- D. S. Johnson and L. A. McGeoch. Experimental analysis of heuristics for the STSP. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*, pages 369–443. Kluwer Academic Publishers, 2002.
- K. Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126:106–130, 2000.
- D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Finding tours in the TSP. Technical Report 99885, Forschungsinstitut für Diskrete Mathematik, University of Bonn, Germany, 1999.
- P. Merz and B. Freisleben. Memetic algorithms for the traveling salesman problem. *Complex Systems*, 4:297–345, 2001.