合肥學院
HEFEI UNIVERSITY

Metaheuristic Optimization
20. Memetic Algorithms

Thomas Weise · 汤卫思

tweise@hfuu.edu.cn · http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Faculty of Computer Science and Technology
Institute of Applied Optimization
230601 Shushan District, Hefei, Anhui, China
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99

合肥学院 南艳湖校区/南2区
计算机科学与技术系
应用优化研究所
中国 安徽省 合肥市 蜀山区 230601
经济技术开发区 锦绣大道99号

Outline

website

Section Outline

Introduction

- We have learned: good optimization requires a balance between

Introduction

- We have learned: good optimization requires a balance between
  - exploitation: search close neighborhood of current best solution – large chance to make small improvements

Introduction

- We have learned: good optimization requires a balance between
  - exploitation: search close neighborhood of current best solution – large chance to make small improvements
  - exploration: search far away, make big search steps – small chance to make large improvements

Introduction

- We have learned: good optimization requires a balance between
  - exploitation: search close neighborhood of current best solution – large chance to make small improvements
  - exploration: search far away, make big search steps – small chance to make large improvements
- This is a fundamental issue *inside* an optimization process

Introduction

- We have learned: good optimization requires a balance between
  - exploitation: search close neighborhood of current best solution – large chance to make small improvements
  - exploration: search far away, make big search steps – small chance to make large improvements
- This is a fundamental issue *inside* an optimization process
- But from the *No Free Lunch Theorem* [1], we can deduce another fundamental dichotomy

Introduction

- We have learned: good optimization requires a balance between
  - exploitation: search close neighborhood of current best solution – large chance to make small improvements
  - exploration: search far away, make big search steps – small chance to make large improvements
- This is a fundamental issue *inside* an optimization process
- But from the *No Free Lunch Theorem* [1], we can deduce another fundamental dichotomy:
  - specialized algorithms are very good for a specific problem, but useless for other problems

Introduction

- We have learned: good optimization requires a balance between
  - exploitation: search close neighborhood of current best solution – large chance to make small improvements
  - exploration: search far away, make big search steps – small chance to make large improvements
- This is a fundamental issue *inside* an optimization process
- But from the *No Free Lunch Theorem* [1], we can deduce another fundamental dichotomy:
  - specialized algorithms are very good for a specific problem, but useless for other problems
  - general algorithms are good for a general class of problems, but not as good as specialized ones for their specific area

Introduction

- Global optimization methods like EAs are general algorithms.

Introduction

- Global optimization methods like EAs are general algorithms.
- They are good at global search, i.e., finding the global optima of different problems

Introduction

- Global optimization methods like EAs are general algorithms.
- They are good at global search, i.e., finding the global optima of different problems *eventually*

Introduction

- Global optimization methods like EAs are general algorithms.
- They are good at global search, i.e., finding the global optima of different problems *eventually*
- They can solve general classes of optimization problems and are usually not specialized

Introduction

- Global optimization methods like EAs are general algorithms.
- They are good at global search, i.e., finding the global optima of different problems *eventually*
- They can solve general classes of optimization problems and are usually not specialized
- Local search algorithms like hill climbers are good at refining an existing solution and quickly tracing down to the bottom of a local optimum

Introduction

- Global optimization methods like EAs are general algorithms.
- They are good at global search, i.e., finding the global optima of different problems *eventually*
- They can solve general classes of optimization problems and are usually not specialized
- Local search algorithms like hill climbers are good at refining an existing solution and quickly tracing down to the bottom of a local optimum
- It is relatively easy to adapt them to a special, very narrow problem, by incorporating complex data structures and operators [2]

Introduction

- Global optimization methods like EAs are general algorithms.
- They are good at global search, i.e., finding the global optima of different problems *eventually*
- They can solve general classes of optimization problems and are usually not specialized
- Local search algorithms like hill climbers are good at refining an existing solution and quickly tracing down to the bottom of a local optimum
- It is relatively easy to adapt them to a special, very narrow problem, by incorporating complex data structures and operators [2]
- How good are general global optimization algorithms compared to specialized local search?

Example: Traveling Salesman Problem

- Let's look on a specific problem class: the TSP [3–6]

w(a,b)	Shanghai	Hefei	Guangzhou	Chengdu
Beijing	1244 km	1044 km	2174 km	1854 km
Chengdu	2095 km	1615 km	1954 km	
Guangzhou	1529 km	1257 km		
Hefei	472 km			

Example: Traveling Salesman Problem

- Let's look on a specific problem class: the TSP [3–6]
- A salesman wants to visit n cities in the shortest possible time. No city should be visited twice and that he wants arrive back at the origin by the end of the tour.

w(a,b)	Shanghai	Hefei	Guangzhou	Chengdu
Beijing	1244 km	1044 km	2174 km	1854 km
Chengdu	2095 km	1615 km	1954 km	
Guangzhou	1529 km	1257 km		
Hefei	472 km			

Example: Traveling Salesman Problem

- Let's look on a specific problem class: the TSP [3–6]
- A salesman wants to visit n cities in the shortest possible time. No city should be visited twice and that he wants arrive back at the origin by the end of the tour.

Definition (Traveling Salesman Problem)

The goal of the Traveling Salesman Problem (TSP) is to find a cyclic path of minimum total weight which visits all vertices of a weighted graph. [3–6]

w(a,b)	Shanghai	Hefei	Guangzhou	Chengdu
Beijing	1244 km	1044 km	2174 km	1854 km
Chengdu	2095 km	1615 km	1954 km	
Guangzhou	1529 km	1257 km		
Hefei	472 km			

Section Outline

Example: Traveling Salesman Problem

w(a,b)	Shanghai	Hefei	Guangzhou	Chengdu
Beijing	1244 km	1044 km	2174 km	1854 km
Chengdu	2095 km	1615 km	1954 km	
Guangzhou	1529 km	1257 km		
Hefei	472 km			

- Problem Space: $\mathbb{X} = \mathbf{\Pi}\{\text{Beijing}, \text{Chengdu}, \text{Guangzhou}, \text{Hefei}, \text{Shanghai}\}$

Example: Traveling Salesman Problem

w(a,b)	Shanghai	Hefei	Guangzhou	Chengdu
Beijing	1244 km	1044 km	2174 km	1854 km
Chengdu	2095 km	1615 km	1954 km	
Guangzhou	1529 km	1257 km		
Hefei	472 km			

- Problem Space: $\mathbb{X} = \mathbf{\Pi}\{\text{Beijing}, \text{Chengdu}, \text{Guangzhou}, \text{Hefei}, \text{Shanghai}\}$
- Objective Function:

Example: Traveling Salesman Problem

w(a,b)	Shanghai	Hefei	Guangzhou	Chengdu
Beijing	1244 km	1044 km	2174 km	1854 km
Chengdu	2095 km	1615 km	1954 km	
Guangzhou	1529 km	1257 km		
Hefei	472 km			

- Problem Space: $\mathbb{X} = \mathbf{\Pi}\{\text{Beijing, Chengdu, Guangzhou, Hefei, Shanghai}\}$
- Objective Function: Minimize $f(x) = \sum_{i=1}^{n} dist(x[i], x[(i+1)\%n])$

Global vs. Local Opt. on the TSP

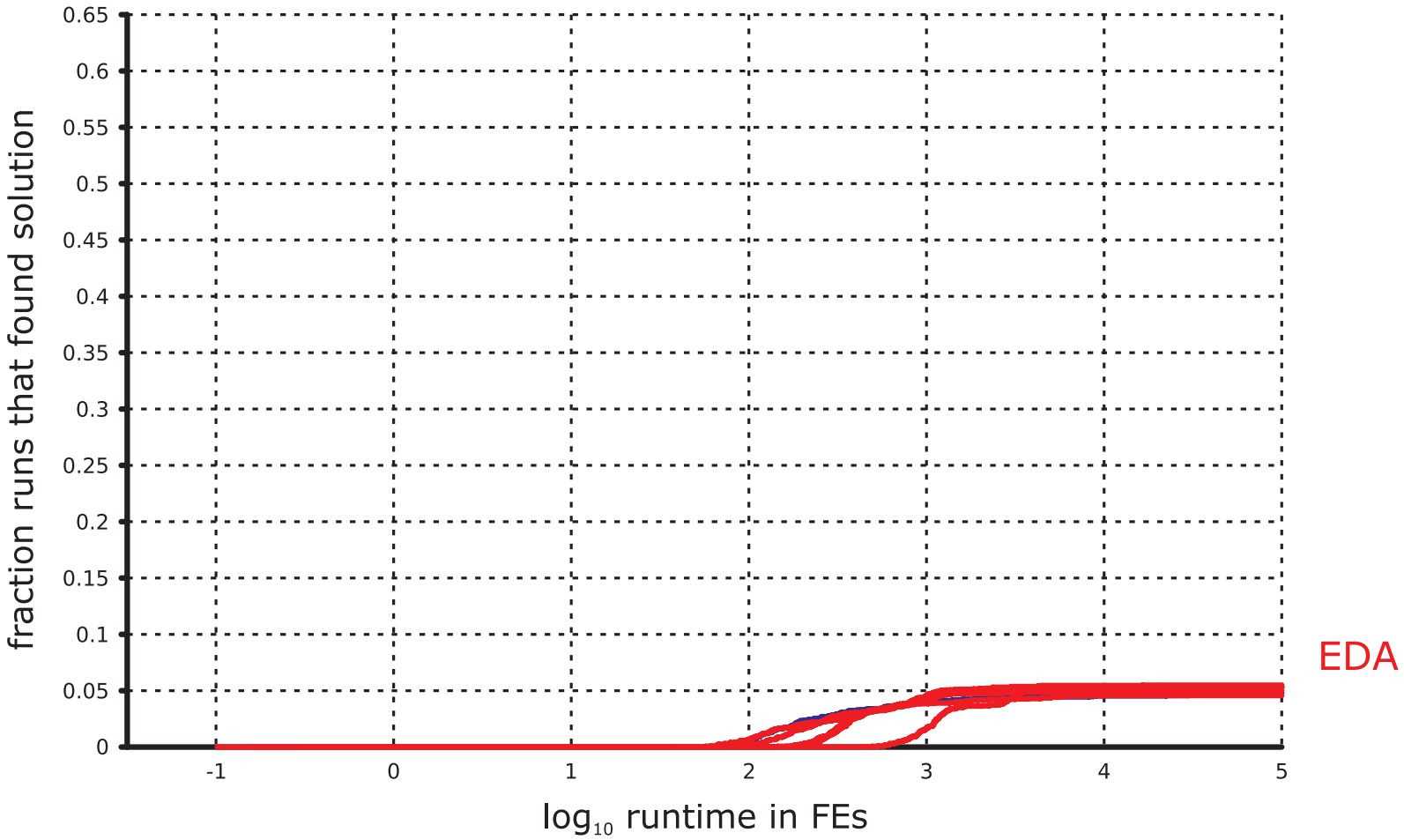
- How good are general global optimization algorithms compared to specialized local search?

Global vs. Local Opt. on the TSP

- How good are general global optimization algorithms compared to specialized local search?
- We apply several algorithms 30 times to each of the 110 symmetric problems from TSPLib [7–10]

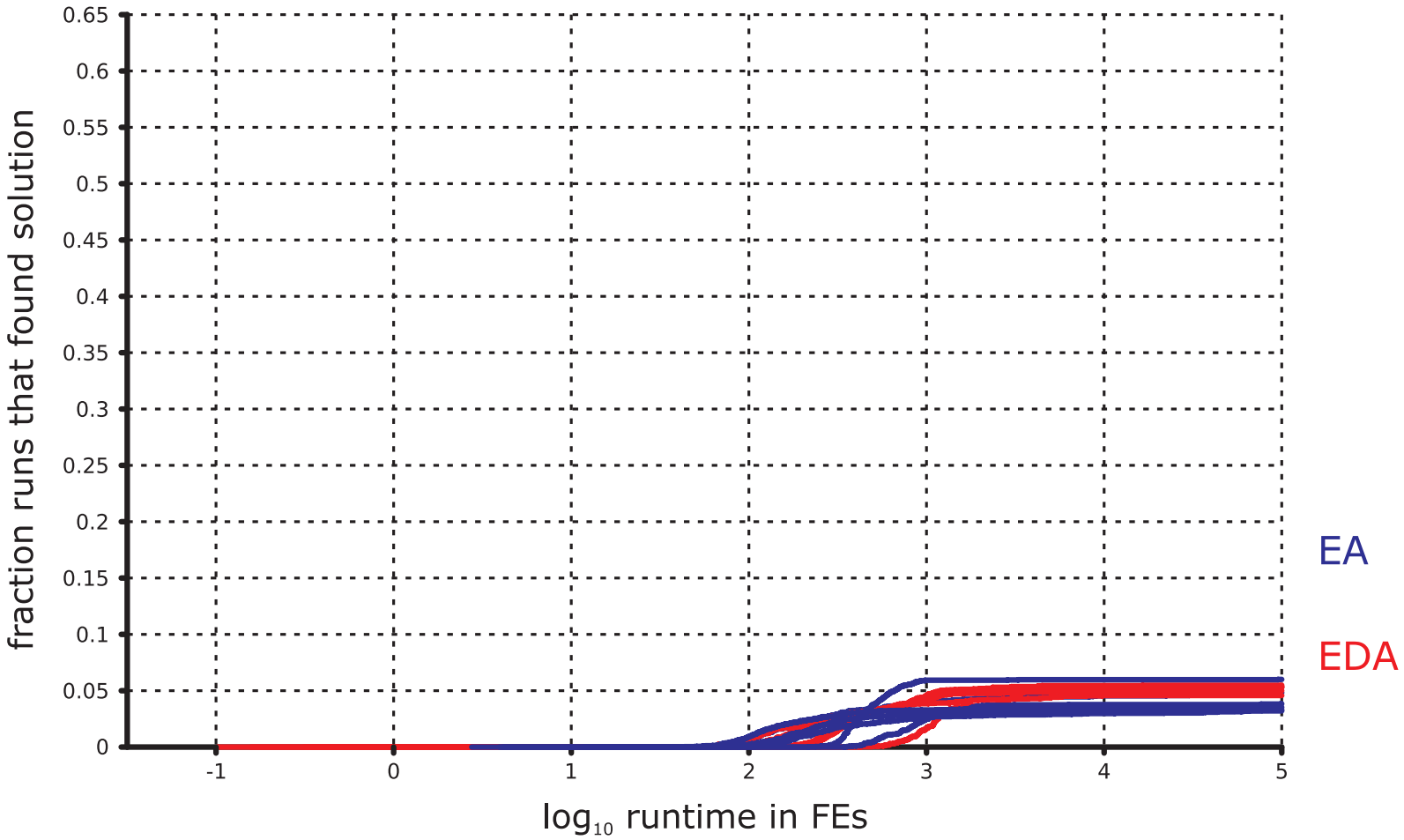
Global vs. Local Opt. on the TSP

- We apply different settings of a good EDA: TEHBSA [11, 12]



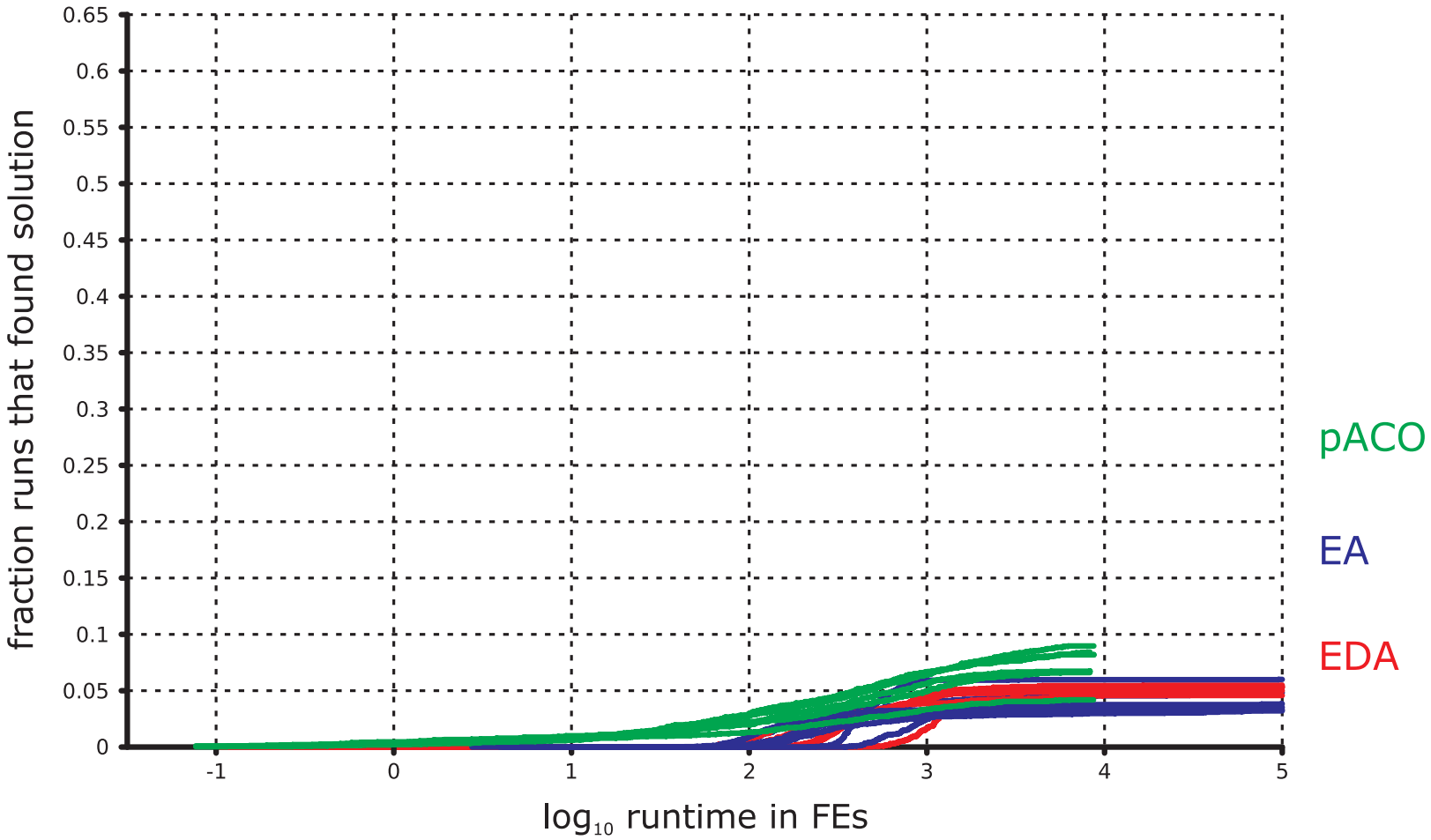
Global vs. Local Opt. on the TSP

- We apply different settings of a good EA [13–16]



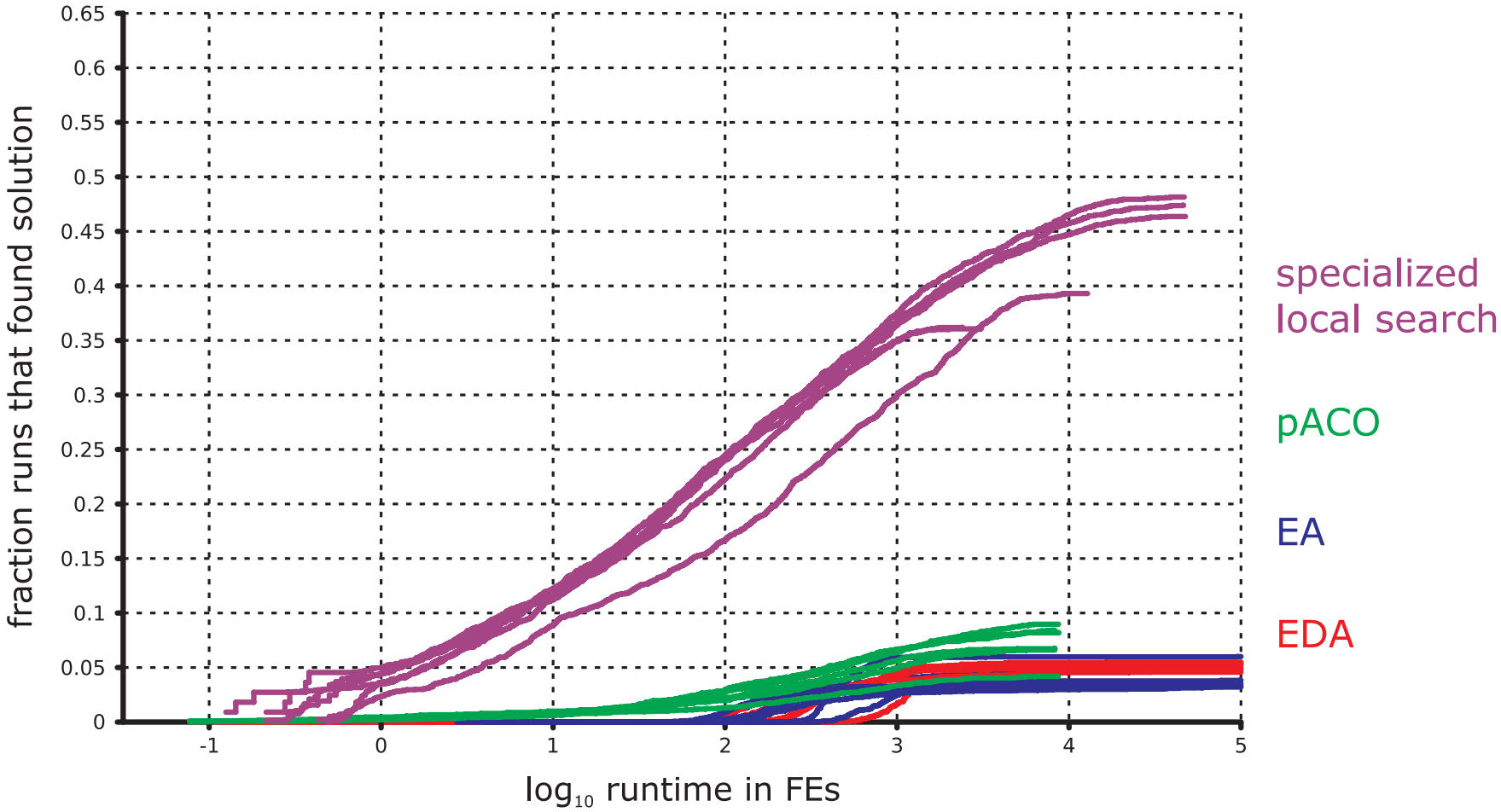
Global vs. Local Opt. on the TSP

- We apply the a good (population-based) ACO [17–19]



Global vs. Local Opt. on the TSP

- . . . and the specialized local search methods (VNS, RNS, MNS) [20–23] . . .



Global vs. Local Opt. on the TSP

- We can design specialized local search methods for this problem that can significantly beat global optimization algorithms!

Global vs. Local Opt. on the TSP

- We can design specialized local search methods for this problem that can significantly beat global optimization algorithms!
- This follows from the No Free Lunch Theorem [1]

Global vs. Local Opt. on the TSP

- We can design specialized local search methods for this problem that can significantly beat global optimization algorithms!
- This follows from the No Free Lunch Theorem [1]
- Of course, we need to understand the problem well, so this is not always possible.
- Does this mean global search is not good for such problems?

Section Outline

Is global optimization useful?

- Does this mean global search is not good for such problems?

Is global optimization useful?

- Does this mean global search is not good for such problems?
- No.

Is global optimization useful?

- Does this mean global search is not good for such problems?
- No.
- Global optimization methods will find the optima, but they are (too) slow

Is global optimization useful?

- Does this mean global search is not good for such problems?
- No.
- Global optimization methods will find the optima, but they are (too) slow
- Local search methods are fast, but may get trapped in local optima

Is global optimization useful?

- Does this mean global search is not good for such problems?
- No.
- Global optimization methods will find the optima, but they are (too) slow
- Local search methods are fast, but may get trapped in local optima
- Let's combine the global search ability of EAs, ACO, and EDAs with the speed of local search!

Is global optimization useful?

- Does this mean global search is not good for such problems?
- No.
- Global optimization methods will find the optima, but they are (too) slow
- Local search methods are fast, but may get trapped in local optima
- Let's combine the global search ability of EAs, ACO, and EDAs with the speed of local search!
- Idea: Always after creating a solution inside the global optimization method, refine it with local search.

Is global optimization useful?

- Does this mean global search is not good for such problems?
- No.
- Global optimization methods will find the optima, but they are (too) slow
- Local search methods are fast, but may get trapped in local optima
- Let's combine the global search ability of EAs, ACO, and EDAs with the speed of local search!
- Idea: Always after creating a solution inside the global optimization method, refine it with local search.
- This concept is called hybridization

Is global optimization useful?

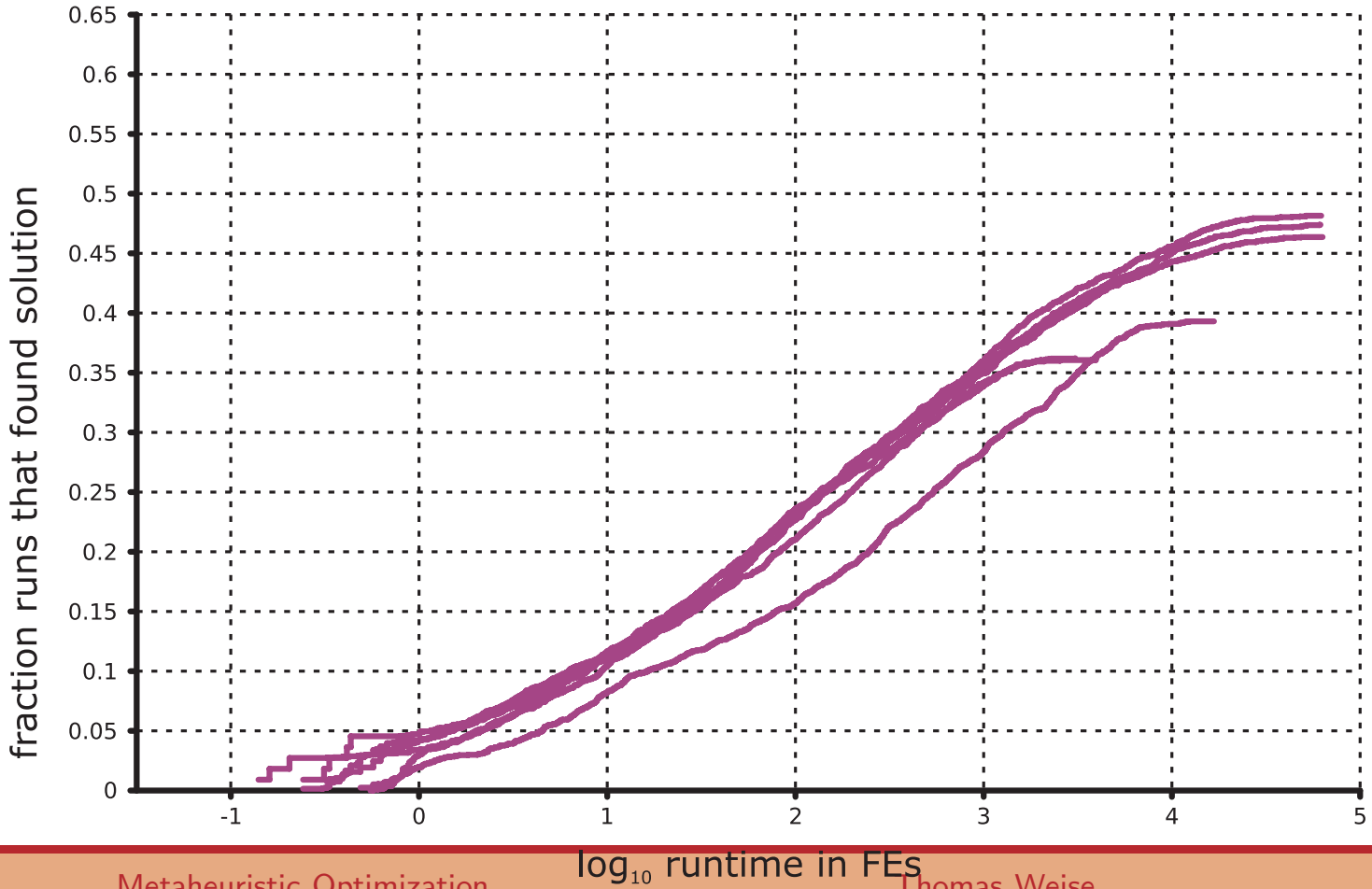
- Does this mean global search is not good for such problems?
- No.
- Global optimization methods will find the optima, but they are (too) slow
- Local search methods are fast, but may get trapped in local optima
- Let's combine the global search ability of EAs, ACO, and EDAs with the speed of local search!
- Idea: Always after creating a solution inside the global optimization method, refine it with local search.
- This concept is called hybridization
- Hybridized EAs are called *Memetic Algorithm* (MA) [24–30]

Memetic Algorithms on TSP

- Let's apply "memetic" versions of the algorithms to the TSP.

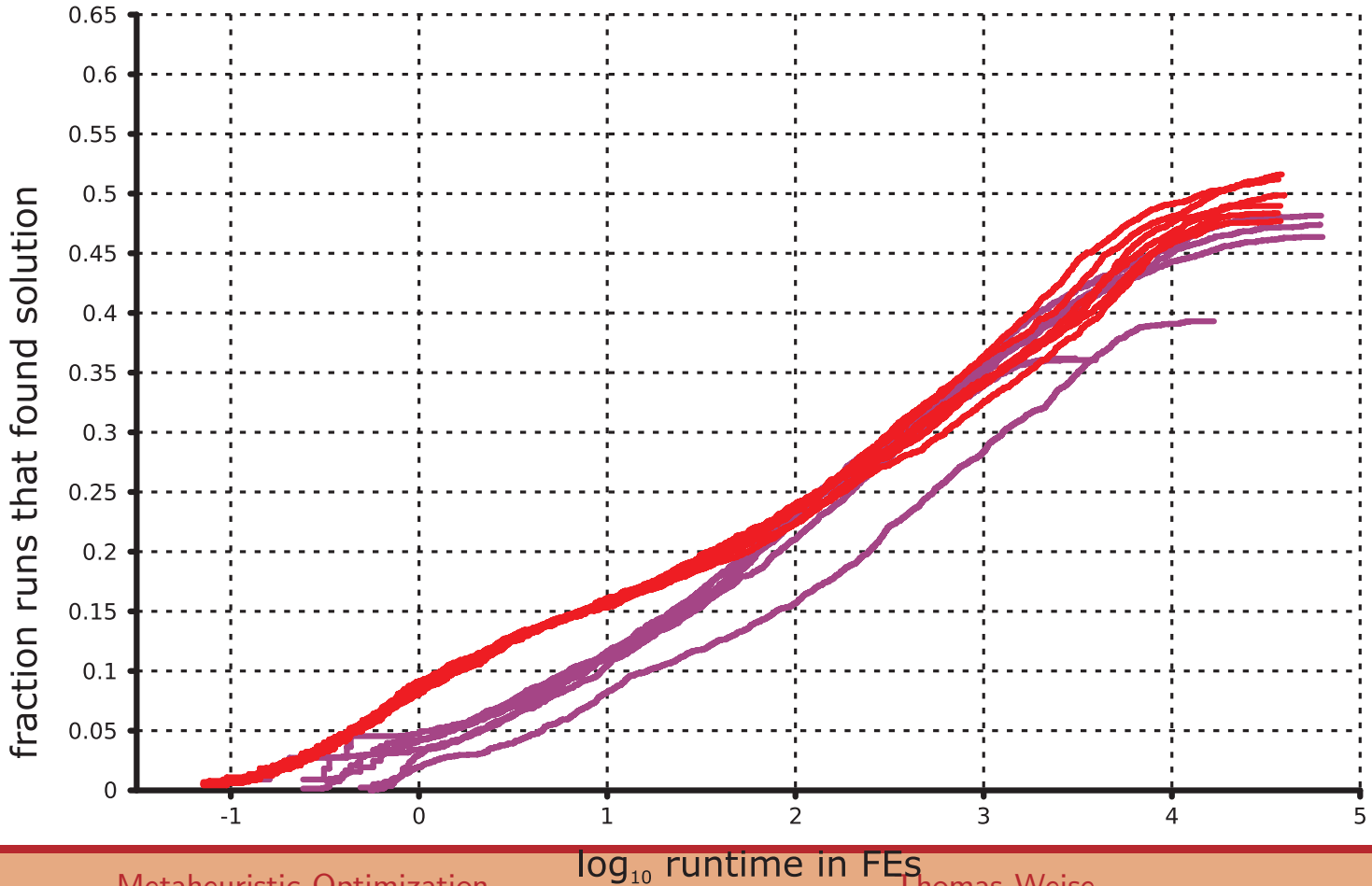
Memetic Algorithms on TSP

- . . . the specialized local search methods (VNS, RNS, MNS) [20–23] . . .



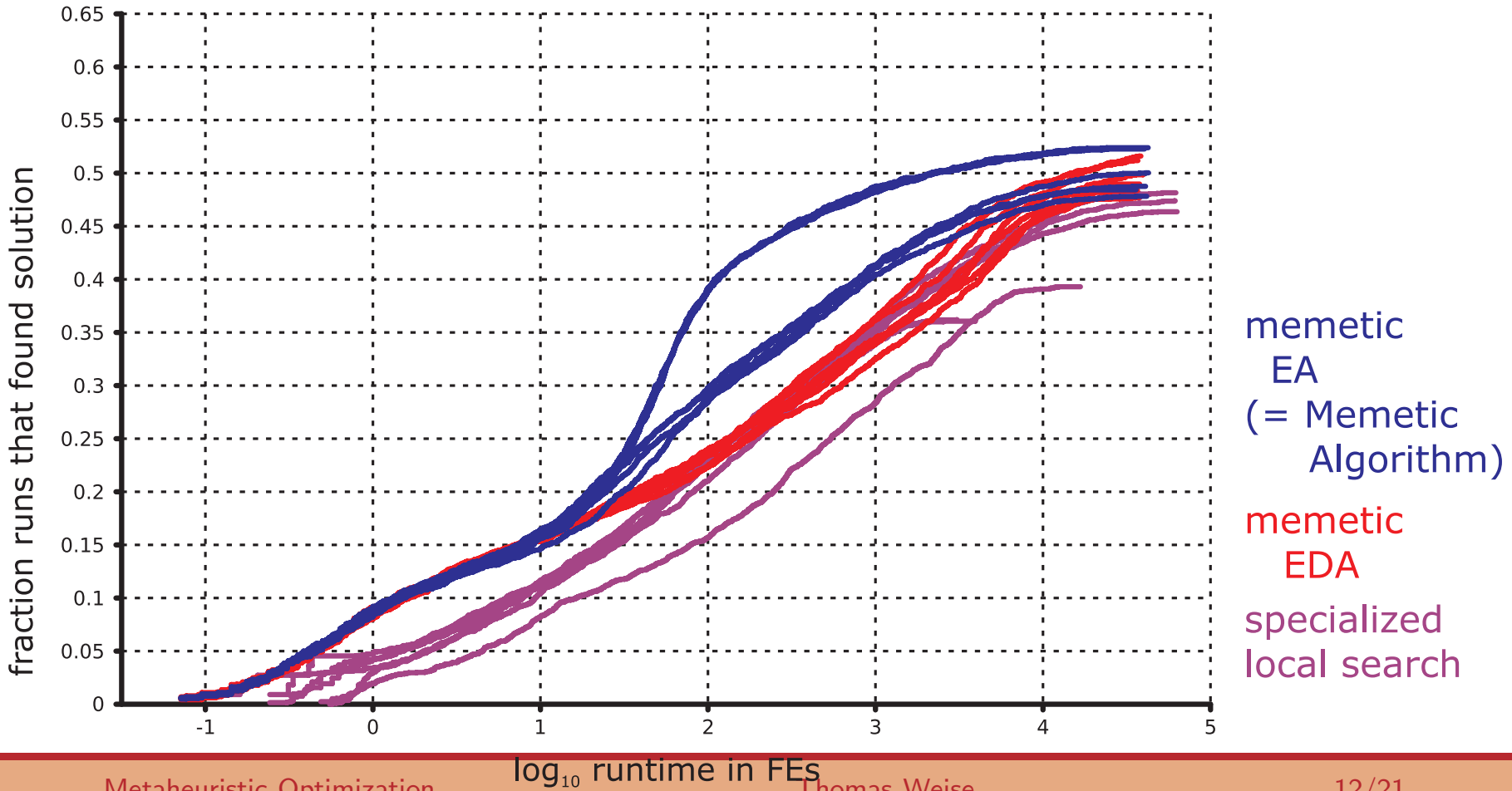
Memetic Algorithms on TSP

- We hybridize the same TEHBSA with MNS



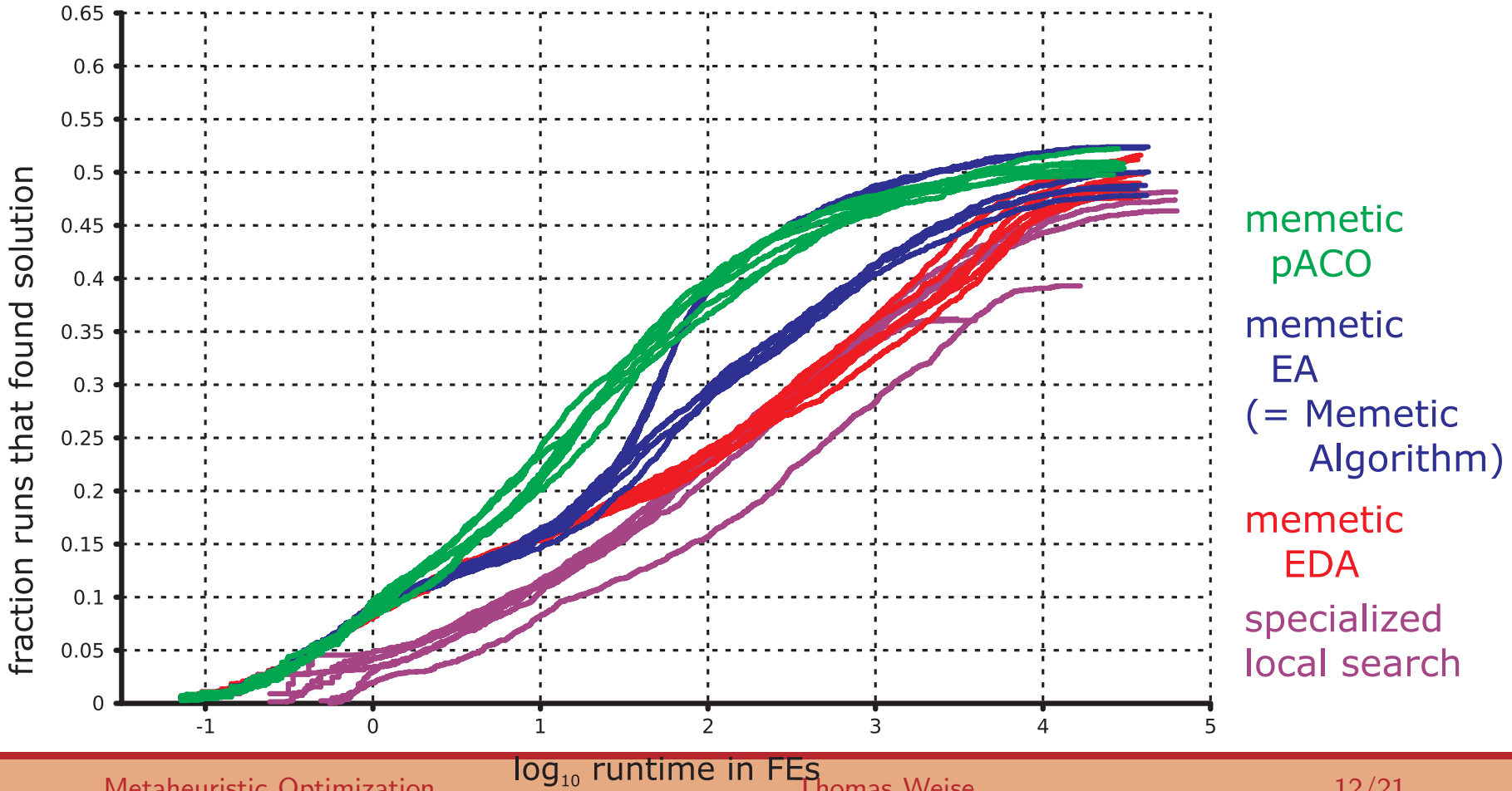
Memetic Algorithms on TSP

- We hybridize the same EA with MNS (the result is called MA)



Memetic Algorithms on TSP

- We hybridize the same pACO with MNS



Procedure

- We take the global search method and implement it exactly as before

Procedure

- We take the global search method and implement it exactly as before
- Then we locate the points where new solutions are created (e.g., mutation, crossover)

Procedure

- We take the global search method and implement it exactly as before
- Then we locate the points where new solutions are created (e.g., mutation, crossover)
- We take these new solutions and use them as starting point for a local search

Procedure

- We take the global search method and implement it exactly as before
- Then we locate the points where new solutions are created (e.g., mutation, crossover)
- We take these new solutions and use them as starting point for a local search, e.g., after each crossover in an EA, we perform a local search

Procedure

- We take the global search method and implement it exactly as before
- Then we locate the points where new solutions are created (e.g., mutation, crossover)
- We take these new solutions and use them as starting point for a local search, e.g., after each crossover in an EA, we perform a local search
- The local search has its own termination criteria

Procedure

- We take the global search method and implement it exactly as before
- Then we locate the points where new solutions are created (e.g., mutation, crossover)
- We take these new solutions and use them as starting point for a local search, e.g., after each crossover in an EA, we perform a local search
- The local search has its own termination criteria, e.g., no improvement for at least 100 steps, or it just scanes one neighborhood completely

Procedure

- We take the global search method and implement it exactly as before
- Then we locate the points where new solutions are created (e.g., mutation, crossover)
- We take these new solutions and use them as starting point for a local search, e.g., after each crossover in an EA, we perform a local search
- The local search has its own termination criteria, e.g., no improvement for at least 100 steps, or it just scanes one neighborhood completely
- The result of the local search then enters the global search

Procedure

- We take the global search method and implement it exactly as before
- Then we locate the points where new solutions are created (e.g., mutation, crossover)
- We take these new solutions and use them as starting point for a local search, e.g., after each crossover in an EA, we perform a local search
- The local search has its own termination criteria, e.g., no improvement for at least 100 steps, or it just scanes one neighborhood completely
- The result of the local search then enters the global search, e.g., crossover $\Longrightarrow$ local search $\Longrightarrow$ offspring population

Procedure

- We take the global search method and implement it exactly as before
- Then we locate the points where new solutions are created (e.g., mutation, crossover)
- We take these new solutions and use them as starting point for a local search, e.g., after each crossover in an EA, we perform a local search
- The local search has its own termination criteria, e.g., no improvement for at least 100 steps, or it just scanes one neighborhood completely
- The result of the local search then enters the global search, e.g., crossover $\Longrightarrow$ local search $\Longrightarrow$ offspring population
- The rest of global search algorithm stays exactly the same

Procedure

- We take the global search method and implement it exactly as before
- Then we locate the points where new solutions are created (e.g., mutation, crossover)
- We take these new solutions and use them as starting point for a local search, e.g., after each crossover in an EA, we perform a local search
- The local search has its own termination criteria, e.g., no improvement for at least 100 steps, or it just scanes one neighborhood completely
- The result of the local search then enters the global search, e.g., crossover $\Longrightarrow$ local search $\Longrightarrow$ offspring population
- The rest of global search algorithm stays exactly the same, e.g., an EA would then perform selection and apply crossover to the offsprings

Procedure

- We take the global search method and implement it exactly as before
- Then we locate the points where new solutions are created (e.g., mutation, crossover)
- We take these new solutions and use them as starting point for a local search, e.g., after each crossover in an EA, we perform a local search
- The local search has its own termination criteria, e.g., no improvement for at least 100 steps, or it just scanes one neighborhood completely
- The result of the local search then enters the global search, e.g., crossover $\Longrightarrow$ local search $\Longrightarrow$ offspring population
- The rest of global search algorithm stays exactly the same, e.g., an EA would then perform selection and apply crossover to the offsprings (and then a local search to each offspring)

Section Outline

Summary

- EAs, EDAs, ACO, PSO, etc. are general algorithms that can be applied without much understanding of the problem

Summary

- EAs, EDAs, ACO, PSO, etc. are general algorithms that can be applied without much understanding of the problem
- They have a good chance to find the global optimum, but they are often slow

Summary

- EAs, EDAs, ACO, PSO, etc. are general algorithms that can be applied without much understanding of the problem
- They have a good chance to find the global optimum, but they are often slow
- Local search methods are specialized and require a good understanding of the problem

Summary

- EAs, EDAs, ACO, PSO, etc. are general algorithms that can be applied without much understanding of the problem
- They have a good chance to find the global optimum, but they are often slow
- Local search methods are specialized and require a good understanding of the problem
- Local search methods are fast, but may get trapped in a local optimum

Summary

- EAs, EDAs, ACO, PSO, etc. are general algorithms that can be applied without much understanding of the problem
- They have a good chance to find the global optimum, but they are often slow
- Local search methods are specialized and require a good understanding of the problem
- Local search methods are fast, but may get trapped in a local optimum
- We can *hybridize* the algorithms: After creating solutions in the main loop of a global search method, we refine them with local searches

Summary

- EAs, EDAs, ACO, PSO, etc. are general algorithms that can be applied without much understanding of the problem
- They have a good chance to find the global optimum, but they are often slow
- Local search methods are specialized and require a good understanding of the problem
- Local search methods are fast, but may get trapped in a local optimum
- We can *hybridize* the algorithms: After creating solutions in the main loop of a global search method, we refine them with local searches
- We obtain Memetic Algorithms, which are fast and good in finding the global optima

谢谢

Thank you

Thomas Weise [汤卫思]
tweise@hfuu.edu.cn
http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Institute of Applied Optimization
Shushan District, Hefei, Anhui,
China

Caspar David Friedrich, “Der Wanderer über dem Nebelmeer”, 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog

Bibliography

Bibliography I

1. David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, 1(1):67–82, April 1997. doi: 10.1109/4235.585893. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.39.6926.
2. Holger H. Hoos and Thomas Stützle. *Stochastic Local Search: Foundations and Applications*. The Morgan Kaufmann Series in Artificial Intelligence. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005. ISBN 1558608729 and 978-1558608726. URL http://books.google.de/books?id=3HAedXnC49IC.
3. Bernhard Friedrich Voigt. *Der Handlungsreisende – wie er sein soll und was er zu thun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiß zu sein – von einem alten Commis-Voyageur*. Ilmenau, Germany: Voigt, 1832. Excerpt: "... Durch geeignete Auswahl und Planung der Tour kann man oft so viel Zeit sparen, daß wir einige Vorschläge zu machen haben. ... Der wichtigste Aspekt ist, so viele Orte wie möglich zu erreichen, ohne einen Ort zweimal zu besuchen. ...".
4. David Lee Applegate, Robert E. Bixby, Vašek Chvátal, and William John Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton Series in Applied Mathematics. Princeton, NJ, USA: Princeton University Press, February 2007. ISBN 0-691-12993-2 and 978-0-691-12993-8. URL http://books.google.de/books?id=nmF4rVNJMVsC.
5. Federico Greco, editor. *Traveling Salesman Problem*. Vienna, Austria: IN-TECH Education and Publishing, September 2008. ISBN 978-953-7619-10-7. URL http://intechweb.org/downloadfinal.php?is=978-953-7619-10-7&type=B.
6. Eugene Leighton (Gene) Lawler, Jan Karel Lenstra, Alexander Hendrik George Rinnooy Kan, and David B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Estimation, Simulation, and Control – Wiley-Interscience Series in Discrete Mathematics and Optimization. Chichester, West Sussex, UK: Wiley Interscience, September 1985. ISBN 0-471-90413-9 and 978-0-471-90413-7. URL http://books.google.de/books?id=BXBGAAAAYAAJ.
7. Gerhard Reinelt. Tsplib, 1995. URL http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.
8. Gerhard Reinelt. Tsplib 95. Technical report, Heidelberg, Germany: Universität Heidelberg, Institut für Mathematik, 1995. URL http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/DOC.PS.
9. Gerhard Reinelt. Tsplib – a traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, Fall 1991. doi: 10.1287/ijoc.3.4.376.
10. William John Cook. Results of concorde for tsplib benchmark, December 2003. URL http://www.tsp.gatech.edu/concorde/benchmarks/bench99.html.

Bibliography II

11. Shigeyoshi Tsutsui. Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram. In Juan Julián Merelo-Guervós, Panagiotis Adamidis, Hans-Georg Beyer, José Luis Fernández-Villacañas Martín, and Hans-Paul Schwefel, editors, *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN VII)*, volume 2439/2002 of *Lecture Notes in Computer Science (LNCS)*, pages 224–233, Granada, Spain, September 7–11, 2002. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/3-540-45712-7_22. URL http://www2.hannan-u.ac.jp/~tsutsui/ps/ppsn2002.pdf.
12. Shigeyoshi Tsutsui. Parallelization of an evolutionary algorithm on a platform with multi-core processors. In Pierre Collet, Nicolas Monmarché, Pierrick Legrand, Marc Schoenauer, and Evelyne Lutton, editors, *Artificial Evolution: Revised Selected Papers from the 9th International Conference, Evolution Artificielle (EA'09)*, volume 5975 of *Lecture Notes in Computer Science (LNCS)*, pages 61–73, Strasbourg, France, October 26–28, 2009. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/978-3-642-14156-0_6.
13. L. Darrell Whitley, Timothy Starkweather, and D'Ann Fuquay. Scheduling problems and traveling salesman: The genetic edge recombination operator. In James David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms (ICGA'89)*, pages 133–140, Fairfax, VA, USA: George Mason University (GMU), June 4–7, 1989. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
14. L. Darrell Whitley, Timothy Starkweather, and Daniel Shaner. The travelling salesman and sequence scheduling: Quality solutions using genetic edge recombination. In Lawrence Davis, editor, *Handbook of Genetic Algorithms*, VNR Computer Library, pages 350–372. Stamford, CT, USA: Thomson Publishing Group, Inc. and New York, NY, USA: Van Nostrand Reinhold Co., January 1991.
15. Timothy Starkweather, S. McDaniel, Keith E. Mathias, L. Darrell Whitley, and C. Whitley. A comparison of genetic sequencing operators. In Richard K. Belew and Lashon Bernard Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA'91)*, pages 69–76, San Diego, CA, USA: University of California (UCSD), July 13–16, 1991. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.4329.
16. Pedro Larrañaga, Cindy M. H. Kuijpers, Roberto H. Murga, Iñaki Inza, and Sejla Dizdarevic. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Journal of Artificial Intelligence Research (JAIR)*, 13(2):129–170, April 1999. doi: 10.1023/A:1006529012972. URL http://www.dca.fee.unicamp.br/~gomide/courses/EA072/artigos/Genetic_Algorithm_TSPR_eview_Larranaga_1999.pdf

Bibliography III

17. Michael Guntsch. *Ant Algorithms in Stochastic and Multi-Criteria Environments*. PhD thesis, Karlsruhe, Germany: University of Karlsruhe (Friedriciana), Department of Economics and Business Engineering and Karlsruhe, Germany: University of Karlsruhe (Friedriciana), Institute for Applied Computer Science and Formal Description Methods (AIFB), January 13, 2004. URL http://www.lania.mx/~ccoello/EMOO/thesis_guntsch.pdf.gz.
18. Michael Guntsch and Martin Middendorf. A population based approach for aco. In Stefano Cagnoni, Jens Gottlieb, Emma Hart, Martin Middendorf, and Günther R. Raidl, editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2002: EvoCOP, EvoIASP, EvoSTIM/EvoPLAN (EvoWorkshops'02)*, volume 2279 of *Lecture Notes in Computer Science (LNCS)*, pages 72–81, Kinsale, Ireland, April 2–4, 2002. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/3-540-46004-7_8. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.2514.
19. Michael Guntsch and Martin Middendorf. Applying population based aco to dynamic optimization problems. In Marco Dorigo, Gianni A. Di Caro, and Michael Samples, editors, *From Ant Colonies to Artificial Ants – Proceedings of the Third International Workshop on Ant Colony Optimization (ANTS'02)*, volume 2463/2002 of *Lecture Notes in Computer Science (LNCS)*, pages 111–122, Brussels, Belgium, September 12–14, 2002. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/3-540-45724-0_10. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.6580.
20. Pierre Hansen and Nenad Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research (EJOR)*, 130(3):449–467, May 1, 2001. doi: 10.1016/S0377-2217(00)00100-4.
21. Pierre Hansen, Nenad Mladenović, and José Andrés Moreno Pérez. Variable neighbourhood search: Methods and applications. *4OR*, 6(4):319–360, December 1, 2008. doi: 10.1007/s10288-008-0089-1.
22. Pierre Hansen, Nenad Mladenović, and José Andrés Moreno Pérez. Variable neighbourhood search: Methods and applications. *Annals of Operations Research*, 175(1):367–407, March 1, 2010. doi: 10.1007/s10479-009-0657-6.
23. Pierre Hansen, Nenad Mladenović, Jack Brimberg, and José Andrés Moreno Pérez. Variable neighborhood search. In Michel Gendrau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, chapter 3, pages 61–86. Norwell, MA, USA: Kluwer Academic Publishers, Dordrecht, Netherlands: Springer Netherlands, and Boston, MA, USA: Springer US, 2010. doi: 10.1007/978-1-4419-1665-5_3.
24. Pablo Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech Concurrent Computation Program C3P 826, Pasadena, CA, USA: California Institute of Technology (Caltech), Caltech Concurrent Computation Program (C3P), 1989. URL http://www.each.usp.br/sarajane/SubPaginas/arquivos_aulas_IA/memetic.pdf.
25. Pablo Moscato. Memetic algorithms. In Panos M. Pardalos and Mauricio G.C. Resende, editors, *Handbook of Applied Optimization*, chapter 3.6.4, pages 157–167. New York, NY, USA: Oxford University Press, Inc., February 22, 2002.

Bibliography IV

26. Pablo Moscato and Carlos Cotta. A gentle introduction to memetic algorithms. In Fred W. Glover and Gary A. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, chapter 5, pages 105–144. Norwell, MA, USA: Kluwer Academic Publishers, Dordrecht, Netherlands: Springer Netherlands, and Boston, MA, USA: Springer US, 2003. doi: 10.1007/0-306-48056-5_5. URL http://www.lcc.uma.es/~ccottap/papers/handbook03memetic.pdf.
27. Ágoston E. Eiben and James E. Smith. Hybridisation with other techniques: Memetic algorithms. In *Introduction to Evolutionary Computing*, Natural Computing Series, chapter 10, pages 173–188. New York, NY, USA: Springer New York, November 2003.
28. William Eugene Hart, Natalio Krasnogor, and James E. Smith, editors. *Recent Advances in Memetic Algorithms*, volume 166/2005 of *Studies in Fuzziness and Soft Computing*. Berlin, Germany: Springer-Verlag GmbH, 2005. ISBN 3-540-22904-3 and 978-3-540-22904-9. doi: 10.1007/3-540-32363-5. URL http://books.google.de/books?id=LYf7YW4DmkUC.
29. Jason Digalakis and Konstantinos Margaritis. Performance comparison of memetic algorithms. *Journal of Applied Mathematics and Computation*, 158:237–252, October 2004. doi: 10.1016/j.amc.2003.08.115. URL http://www.complexity.org.au/ci/draft/draft/digala02/digala02s.pdf.
30. Nicholas J. Radcliffe and Patrick David Surry. Formal memetic algorithms. In Terence Claus Fogarty, editor, *Proceedings of the Workshop on Artificial Intelligence and Simulation of Behaviour, International Workshop on Evolutionary Computing, Selected Papers (AISB'94)*, volume 865/1994 of *Lecture Notes in Computer Science (LNCS)*, pages 1–16, Leeds, UK, April 11–13, 1994. Chichester, West Sussex, UK: Society for the Study of Artificial Intelligence and the Simulation of Behaviour (SSAISB), Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/3-540-58483-8_1. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.9885.