中国科学技术大学
University of Science and Technology of China

The optimizationBenchmarking.org Experiment Evaluation Framework

Thomas Weise
tweise@ustc.edu.cn · tweise@gmx.de · http://www.it-weise.de

USTC-Birmingham Joint Res. Inst. in Intelligent Computation and Its Applications (UBRI)
University of Science and Technology of China (USTC), Hefei 230027, Anhui, China

April 17, 2015

Outline

- Many questions in the real world are actually optimization problems

- Many questions in the real world are actually optimization problems, e.g.,
  - Find the *shortest* tour for a salesman to visit Beijing, Chengdu, Shanghai, Wuhan, Xianggang, and return to Hefei!

- Many questions in the real world are actually optimization problems, e.g.,
  - Find the *shortest* tour for a salesman to visit Beijing, Chengdu, Shanghai, Wuhan, Xianggang, and return to Hefei!
  - I need to transport n items from here to Feixi but they are too big to transport them all at once. How can I load them best into my car so that I have to travel back and forth the least times?

- Many questions in the real world are actually optimization problems, e.g.,
  - Find the *shortest* tour for a salesman to visit Beijing, Chengdu, Shanghai, Wuhan, Xianggang, and return to Hefei!
  - I need to transport n items from here to Feixi but they are too big to transport them all at once. How can I load them best into my car so that I have to travel back and forth the least times?
  - Which setting of x_1, x_2, x_3, and x_4 can make $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$ become true (or, at least, as *many* of its terms as possible)?

- Many questions in the real world are actually optimization problems, e.g.,
  - Find the *shortest* tour for a salesman to visit Beijing, Chengdu, Shanghai, Wuhan, Xianggang, and return to Hefei!
  - I need to transport n items from here to Feixi but they are too big to transport them all at once. How can I load them best into my car so that I have to travel back and forth the least times?
  - Which setting of x_1, x_2, x_3, and x_4 can make $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$ become true (or, at least, as *many* of its terms as possible)?
  - I want to build a large factory with n workshops. I know the flow of material between each two workshops and now need to choose the locations of the workshops such that the overall running cost incurred by material transportation is *minimized*.

- Many questions in the real world are actually optimization problems, e.g.,
  - Find the *shortest* tour for a salesman to visit Beijing, Chengdu, Shanghai, Wuhan, Xianggang, and return to Hefei!
  - I need to transport n items from here to Feixi but they are too big to transport them all at once. How can I load them best into my car so that I have to travel back and forth the least times?
  - Which setting of x_1, x_2, x_3, and x_4 can make $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$ become true (or, at least, as *many* of its terms as possible)?
  - I want to build a large factory with n workshops. I know the flow of material between each two workshops and now need to choose the locations of the workshops such that the overall running cost incurred by material transportation is *minimized*.
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13]

Optimization Algorithms

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19], Differential Evolution [9]

Optimization Algorithms

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19], Differential Evolution [9], Particle Swarm Optimization [9]

Optimization Algorithms

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19], Differential Evolution [9], Particle Swarm Optimization [9], Estimation of Distribution Algorithms [20–27]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19], Differential Evolution [9], Particle Swarm Optimization [9], Estimation of Distribution Algorithms [20–27], CMA-ES [28–35]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19], Differential Evolution [9], Particle Swarm Optimization [9], Estimation of Distribution Algorithms [20–27], CMA-ES [28–35], and Local Search methods [36–38]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19], Differential Evolution [9], Particle Swarm Optimization [9], Estimation of Distribution Algorithms [20–27], CMA-ES [28–35], and Local Search methods [36–38] such as Simulated Annealing [9, 39–47]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19], Differential Evolution [9], Particle Swarm Optimization [9], Estimation of Distribution Algorithms [20–27], CMA-ES [28–35], and Local Search methods [36–38] such as Simulated Annealing [9, 39–47] or Tabu Search [48–52]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are Evolutionary Algorithms [1–9], Ant Colony Optimization [9–13], Evolution Strategies [9, 14–19], Differential Evolution [9], Particle Swarm Optimization [9], Estimation of Distribution Algorithms [20–27], CMA-ES [28–35], and Local Search methods [36–38] such as Simulated Annealing [9, 39–47] or Tabu Search [48–52], as well as hybrids of local and global search, such as Memetic Algorithms [53–59]

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are... many
- Which of them is best (for my problem)?

Optimization Algorithms

- Many questions in the real world are actually optimization problems, e.g.,
  - Traveling Salesman Problem [60–63]
  - Bin Packing Problem [64]
  - Maximum (3-)Satisfiability Problem [65–68]
  - Quadratic Assignment Problem [69, 70]
- Many optimization problems are $\mathcal{NP}$-hard, meaning that finding the best possible solution will usually not be possible in feasible time.
- We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime.
- Examples of such algorithms are… many
- Which of them is best (for my problem)?
- How can I make a good algorithm better (for my problem)?

- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs* $\mathcal{O}(n \log n)$ *while Bubble Sort needs* $\mathcal{O}(n^2)$ *steps in the average case."*

- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs* $\mathcal{O}(n \log n)$ *while Bubble Sort needs* $\mathcal{O}(n^2)$ *steps in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality

- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs* $\mathcal{O}(n \log n)$ *while Bubble Sort needs* $\mathcal{O}(n^2)$ *steps in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible

- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs* $\mathcal{O}(n \log n)$ *while Bubble Sort needs* $\mathcal{O}(n^2)$ *steps in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible
  - analysis extremely complicated

- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs $\mathcal{O}(n \log n)$ while Bubble Sort needs $\mathcal{O}(n^2)$ steps in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible
  - analysis extremely complicated since
  - algorithms are usually randomized

- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs* $\mathcal{O}(n \log n)$ *while Bubble Sort needs* $\mathcal{O}(n^2)$ *steps in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible
  - analysis extremely complicated since
  - algorithms are usually randomized and
  - have many parameters (e.g., crossover rate, population size)

- Traditional Approach à la *“QuickSort is better than Bubble Sort because it needs* $\mathcal{O}(n \log n)$ *while Bubble Sort needs* $\mathcal{O}(n^2)$ *steps in the average case.”*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible
    - analysis extremely complicated since
    - algorithms are usually randomized and
    - have many parameters (e.g., crossover rate, population size) and
    - “sub-algorithms” (e.g., crossover operator, mutation operator, selection algorithm)

- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs $\mathcal{O}(n \log n)$ while Bubble Sort needs $\mathcal{O}(n^2)$ steps in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible
  - analysis extremely complicated since
  - algorithms are usually randomized and
  - have many parameters (e.g., crossover rate, population size) and
  - "sub-algorithms" (e.g., crossover operator, mutation operator, selection algorithm)
  - optimization problems also differ in many aspects

- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs* $\mathcal{O}(n \log n)$ *while Bubble Sort needs* $\mathcal{O}(n^2)$ *steps in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible
  - analysis extremely complicated since
  - algorithms are usually randomized and
  - have many parameters (e.g., crossover rate, population size) and
  - "sub-algorithms" (e.g., crossover operator, mutation operator, selection algorithm)
  - optimization problems also differ in many aspects
  - Theoretical results only available for toy problems and extremely simplified algorithms.

- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs $\mathcal{O}(n \log n)$ while Bubble Sort needs $\mathcal{O}(n^2)$ steps in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible
  - analysis extremely complicated since
  - algorithms are usually randomized and
  - have many parameters (e.g., crossover rate, population size) and
  - "sub-algorithms" (e.g., crossover operator, mutation operator, selection algorithm)
  - optimization problems also differ in many aspects
  - Theoretical results only available for toy problems and extremely simplified algorithms.
  - Currently, not mature enough to be an easy-to-use tool for practitioners

中国科学技术大学
University of Science and Technology of China

- Traditional Approach à la *"QuickSort is better than Bubble Sort because it needs* $\mathcal{O}(n \log n)$ *while Bubble Sort needs* $\mathcal{O}(n^2)$ *steps in the average case."*
- Complexity Analysis, Theoretical Bounds of Runtime and Solution Quality
- Usually not feasible
  - analysis extremely complicated since
  - algorithms are usually randomized and
  - have many parameters (e.g., crossover rate, population size) and
  - "sub-algorithms" (e.g., crossover operator, mutation operator, selection algorithm)
  - optimization problems also differ in many aspects
  - Theoretical results only available for toy problems and extremely simplified algorithms.
  - Currently, not mature enough to be an easy-to-use tool for practitioners
- Experimental analysis and comparison only practical alternative.

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

- Algorithm performance has two dimensions [71, 72]:

.

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

- Algorithm performance has two dimensions [71, 72]: solution quality

.

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

- Algorithm performance has two dimensions [71, 72]: solution quality and required runtime.

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

- Algorithm performance has two dimensions [71, 72]: solution quality and required runtime.
- Anytime Algorithms [73] are optimization methods which have a guess about what the optimum could be at *any time* during their run and which iteratively improve this guess.

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

- Algorithm performance has two dimensions [71, 72]: solution quality and required runtime.
- Anytime Algorithms [73] are optimization methods which have a guess about what the optimum could be at *any time* during their run and which iteratively improve this guess.
- All metaheuristics are Anytime Algorithms.

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

- Algorithm performance has two dimensions [71, 72]: solution quality and required runtime.
- Anytime Algorithms [73] are optimization methods which have a guess about what the optimum could be at *any time* during their run and which iteratively improve this guess.
- All metaheuristics are Anytime Algorithms.
- Several exact methods like Branch-and-Bound [74–76] are Anytime Algorithms.

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

- Algorithm performance has two dimensions [71, 72]: solution quality and required runtime.
- Anytime Algorithms [73] are optimization methods which have a guess about what the optimum could be at *any time* during their run and which iteratively improve this guess.
- All metaheuristics are Anytime Algorithms.
- Several exact methods like Branch-and-Bound [74–76] are Anytime Algorithms.
- Consequence: Most optimization algorithms produce approximate solutions of different qualities at different points during their process.

"We use metaheuristic optimization algorithms to give us good approximate solutions within acceptable runtime."

- Algorithm performance has two dimensions [71, 72]: solution quality and required runtime.
- Anytime Algorithms [73] are optimization methods which have a guess about what the optimum could be at *any time* during their run and which iteratively improve this guess.
- All metaheuristics are Anytime Algorithms.
- Several exact methods like Branch-and-Bound [74–76] are Anytime Algorithms.
- Consequence: Most optimization algorithms produce approximate solutions of different qualities at different points during their process.
- Experiments must capture solution quality and runtime data.

Structure of the Framework

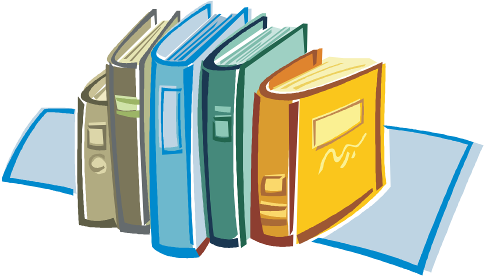
谢谢！

Thank you.

Thomas Weise
tweise@ustc.edu.cn · tweise@gmx.de · http://www.it-weise.de

USTC-Birmingham Joint Res. Inst. in Intelligent Computation and Its Applications (UBRI)
University of Science and Technology of China (USTC), Hefei 230027, Anhui, China

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



1. Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors. *Handbook of Evolutionary Computation*. Computational Intelligence Library. New York, NY, USA: Oxford University Press, Inc., Dirac House, Temple Back, Bristol, UK: Institute of Physics Publishing Ltd. (IOP), and Boca Raton, FL, USA: CRC Press, Inc., January 1, 1997. ISBN 0-7503-0392-1, 0-7503-0895-8, 978-0-7503-0392-7, and 978-0-7503-0895-3. URL http://books.google.de/books?id=n5nuiIZvmpAC.
2. Raymond Chiong, Thomas Weise, and Zbigniew Michalewicz, editors. *Variants of Evolutionary Algorithms for Real-World Applications*. Berlin/Heidelberg: Springer-Verlag, 2011. ISBN 978-3-642-23423-1 and 978-3-642-23424-8. doi: 10.1007/978-3-642-23424-8. URL http://books.google.de/books?id=B2ONePP40MEC.
3. Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors. *Evolutionary Computation 1: Basic Algorithms and Operators*. Dirac House, Temple Back, Bristol, UK: Institute of Physics Publishing Ltd. (IOP), January 2000. ISBN 0750306645 and 9780750306645. URL http://books.google.de/books?id=4HMYCq9US78C.
4. Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors. *Evolutionary Computation 2: Advanced Algorithms and Operators*. Dirac House, Temple Back, Bristol, UK: Institute of Physics Publishing Ltd. (IOP), November 2000. ISBN 0750306653 and 9780750306652.
5. Dumitru (Dan) Dumitrescu, Beatrice Lazzerini, Lakhmi C. Jain, and A. Dumitrescu. *Evolutionary Computation*, volume 18 of *International Series on Computational Intelligence*. Boca Raton, FL, USA: CRC Press, Inc., June 2000. ISBN 0-8493-0588-8 and 978-0-8493-0588-7. URL http://books.google.de/books?id=MSU9ep79JvUC.
6. Ágoston E. Eiben, editor. *Evolutionary Computation*. Theoretical Computer Science. Amsterdam, The Netherlands: IOS Press, 1999. ISBN 4-274-90269-2, 90-5199-471-0, 978-4-274-90269-7, and 978-90-5199-471-1. URL http://books.google.de/books?id=8LVAGQAACAAJ. This is the book edition of the journal Fundamenta Informaticae, Volume 35, Nos. 1-4, 1998.
7. David Wolfe Corne, Marco Dorigo, Fred W. Glover, Dipankar Dasgupta, Pablo Moscato, Riccardo Poli, and Kenneth V. Price, editors. *New Ideas in Optimization*. McGraw-Hill's Advanced Topics In Computer Science Series. Maidenhead, England, UK: McGraw-Hill Ltd., May 1999. ISBN 0-07-709506-5 and 978-0-07-709506-2. URL http://books.google.de/books?id=nC35AAAACAAJ.
8. Ashish Ghosh and Shigeyoshi Tsutsui, editors. *Advances in Evolutionary Computing – Theory and Applications*. Natural Computing Series. New York, NY, USA: Springer New York, November 22, 2002. ISBN 3-540-43330-9 and 978-3-540-43330-9. URL http://books.google.de/books?id=OGMEMC9P3vMC.
9. Thomas Weise. *Global Optimization Algorithms – Theory and Application*. Germany: it-weise.de (self-published), 2009. URL http://www.it-weise.de/projects/book.pdf.

10. Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(1):29–41, February 1996. doi: 10.1109/3477.484436. URL ftp://iridia.ulb.ac.be/pub/mdorigo/journals/IJ.10-SMC96.pdf.
11. Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. Bradford Books. Cambridge, MA, USA: MIT Press, July 1, 2004. ISBN 0-262-04219-3 and 978-0-262-04219-2. URL http://books.google.de/books?id=_aefcpY8GiEC.
12. 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.
13. Mark Zlochin, Mauro Birattari, Nicolas Meuleau, and Marco Dorigo. Model-based search for combinatorial optimization: A critical survey. *Annals of Operations Research*, 132(1-4):373–395, November 2004. doi: 10.1023/B:ANOR.0000039526.52305.af.
14. Ingo Rechenberg. *Cybernetic Solution Path of an Experimental Problem*. Farnborough, Hampshire, UK: Royal Aircraft Establishment, August 1965. Library Translation 1122.
15. Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, Berlin, Germany: Technische Universität Berlin, 1971. URL http://books.google.de/books?id=QcNNGQAACAAJ.
16. Ingo Rechenberg. *Evolutionsstrategie '94*, volume 1 of *Werkstatt Bionik und Evolutionstechnik*. Bad Cannstadt, Stuttgart, Baden-Württemberg, Germany: Frommann-Holzboog Verlag, 1994. ISBN 3-7728-1642-8 and 978-3-772-81642-0. URL http://books.google.de/books?id=savAAAACAAJ.
17. Hans-Paul Schwefel. Kybernetische evolution als strategie der exprimentellen forschung in der strömungstechnik. Master's thesis, Berlin, Germany: Technische Universität Berlin, 1965.
18. Hans-Paul Schwefel. Experimentelle optimierung einer zweiphasendüse teil i. Technical Report 35, Berlin, Germany: AEG Research Institute, 1968. Project MHD–Staustrahlrohr 11.034/68.
19. Hans-Paul Schwefel. *Evolutionsstrategie und numerische Optimierung*. PhD thesis, Berlin, Germany: Technische Universität Berlin, Institut für Meß- und Regelungstechnik, Institut für Biologie und Anthropologie, 1975.
20. Kenneth V. Price, Rainer M. Storn, and Jouni A. Lampinen. *Differential Evolution – A Practical Approach to Global Optimization*. Natural Computing Series. Basel, Switzerland: Birkhäuser Verlag, 2005. ISBN 3-540-20950-6, 3-540-31306-0, 978-3-540-20950-8, and 978-3-540-31306-9. URL http://books.google.de/books?id=S67vX-KqVqUC.

21. Vitaliy Feoktistov. *Differential Evolution – In Search of Solutions*, volume 5 of *Springer Optimization and Its Applications*. New York, NY, USA: Springer New York, December 2006. ISBN 0-387-36895-7, 0-387-36896-5, 978-0-387-36895-5, and 978-0-387-36896-2. URL `http://books.google.de/books?id=kG7aP_v-SU4C`.
22. Efrén Mezura-Montes, Jesús Velázquez-Reyes, and Carlos Artemio Coello Coello. A comparative study of differential evolution variants for global optimization. In Maarten Keijzer and Mike Cattolico, editors, *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'06)*, pages 485–492, Seattle, WA, USA: Renaissance Seattle Hotel, July 8–12, 2006. New York, NY, USA: ACM Press. doi: 10.1145/1143997.1144086. URL `http://delta.cs.cinvestav.mx/~ccoello/conferences/mezura-gecco2006.pdf.gz`.
23. Janez Brest, Viljem Žumer, and Mirjam Sepesy Maučec. Control parameters in self-adaptive differential evolution. In Bogdan Filipič and Jurij Šilc, editors, *Proceedings of the Second International Conference on Bioinspired Optimization Methods and their Applications (BIOMA'06)*, Informacijska Družba (Information Society), pages 35–44, Ljubljana, Slovenia: Jožef Stefan International Postgraduate School, October 9–10, 2006. Ljubljana, Slovenia: Jožef Stefan Institute. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.106.8106`.
24. Jouni A. Lampinen and Ivan Zelinka. On stagnation of the differential evolution algorithm. In Pavel Osmera, editor, *Proceedings of the 6th International Conference on Soft Computing (MENDEL'00)*, pages 76–83, Brno, Czech Republic: Brno University of Technology, June 7–9, 2000. Brno, Czech Republic: Brno University of Technology, Ústav Automatizace a Informatiky. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.7932`.
25. Roberto R. F. Mendes and Arvind S. Mohais. Dynde: A differential evolution for dynamic optimization problems. In David Wolfe Corne, Zbigniew Michalewicz, Robert Ian McKay, Ágoston E. Eiben, David B. Fogel, Carlos M. Fonseca, Günther R. Raidl, Kay Chen Tan, and Ali M. S. Zalzala, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*, volume 3, pages 2808–2815, Edinburgh, Scotland, UK, September 2–5, 2005. Piscataway, NJ, USA: IEEE Computer Society. doi: 10.1109/CEC.2005.1555047. URL `http://www3.di.uminho.pt/~rcm/publications/DynDE.pdf`.
26. Patricia Besson, Jean-Marc Vesin, Vlad Popovici, and Murat Kunt. Differential evolution applied to a multimodal information theoretic optimization problem. In Franz Rothlauf, Jürgen Branke, Stefano Cagnoni, Ernesto Jorge Fernandes Costa, Carlos Cotta, Rolf Drechsler, Evelyne Lutton, Penousal Machado, Jason H. Moore, Juan Romero, George D. Smith, Giovanni Squillero, and Hideyuki Takagi, editors, *Applications of Evolutionary Computing – Proceedings of EvoWorkshops 2006: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, and EvoSTOC (EvoWorkshops'06)*, volume 3907/2006 of *Lecture Notes in Computer Science (LNCS)*, pages 505–509, Budapest, Hungary, April 10–12, 2006. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/11732242_46.

27. Rainer M. Storn. Differential evolution (de) for continuous function optimization (an algorithm by kenneth price and rainer storn), 2010. URL http://www.icsi.berkeley.edu/~storn/code.html.
28. Nikolaus Hansen, Andreas Ostermeier, and Andreas Gawelczyk. On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA'95)*, pages 57–64, Pittsburgh, PA, USA: University of Pittsburgh, July 15–19, 1995. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.9321.
29. Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In Keisoku Jidō and Seigyo Gakkai, editors, *Proceedings of IEEE International Conference on Evolutionary Computation (CEC'96)*, pages 312–317, Nagoya, Aichi, Japan: Nagoya University, Symposium & Toyoda Auditorium, May 20–22, 1996. Los Alamitos, CA, USA: IEEE Computer Society Press. URL http://www.lri.fr/~hansen/CMAES.pdf.
30. Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. URL http://www.bionik.tu-berlin.de/user/niko/cmaartic.pdf.
31. Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 11(1):1–18, Spring 2003. doi: 10.1162/106365603321828970. URL http://mitpress.mit.edu/journals/pdf/evco_11_1_1_0.pdf.
32. Nikolaus Hansen and Stefan Kern. Evaluating the cma evolution strategy on multimodal test functions. In Xin Yao, Edmund K. Burke, José Antonio Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiño, Ata Kabán, and Hans-Paul Schwefel, editors, *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242/2004 of *Lecture Notes in Computer Science (LNCS)*, pages 282–291, Birmingham, UK, September 18–22, 2008. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/978-3-540-30217-9_29. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.69.163.
33. Nikolaus Hansen. The cma evolution strategy: A comparing review. In José Antonio Lozano, Pedro Larrañaga, Iñaki Inza, and Endika Bengoetxea, editors, *Towards a New Evolutionary Computation – Advances on Estimation of Distribution Algorithms*, volume 192/2006 of *Studies in Fuzziness and Soft Computing*, pages 75–102. Berlin, Germany: Springer-Verlag GmbH, 2006. URL http://www.lri.fr/~hansen/hansenedacomparing.pdf.

34. Anne Auger and Nikolaus Hansen. A restart cma evolution strategy with increasing population size. In David Wolfe Corne, Zbigniew Michalewicz, Robert Ian McKay, Ágoston E. Eiben, David B. Fogel, Carlos M. Fonseca, Günther R. Raidl, Kay Chen Tan, and Ali M. S. Zalzala, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*, pages 1769–1776, Edinburgh, Scotland, UK, September 2–5, 2005. Piscataway, NJ, USA: IEEE Computer Society. doi: 10.1109/CEC.2005.1554902. URL `http://www.lri.fr/~hansen/cec2005ipopcmaes.pdf`.
35. Anne Auger and Nikolaus Hansen. Performance evaluation of an advanced local search evolutionary algorithm. In David Wolfe Corne, Zbigniew Michalewicz, Robert Ian McKay, Ágoston E. Eiben, David B. Fogel, Carlos M. Fonseca, Günther R. Raidl, Kay Chen Tan, and Ali M. S. Zalzala, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*, volume 2, pages 1777–1784, Edinburgh, Scotland, UK, September 2–5, 2005. Piscataway, NJ, USA: IEEE Computer Society. doi: 10.1109/CEC.2005.1554903. URL `http://www.lri.fr/~hansen/cec2005localcmaes.pdf`.
36. 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`.
37. Emile H. L. Aarts and Jan Karel Lenstra, editors. *Local Search in Combinatorial Optimization*. Estimation, Simulation, and Control – Wiley-Interscience Series in Discrete Mathematics and Optimization. Princeton, NJ, USA: Princeton University Press, 1997. ISBN 0585277540, 0691115222, 9780585277547, and 9780691115221. URL `http://books.google.de/books?id=NWghN9G7q9MC`.
38. Matthijs den Besten, Thomas Stützle, and Marco Dorigo. Design of iterated local search algorithms. In Egbert J. W. Boers, Jens Gottlieb, Pier Luca Lanzi, Robert Elliott Smith, Stefano Cagnoni, Emma Hart, Günther R. Raidl, and Harald Tijink, editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2001: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM (EvoWorkshops'01)*, volume 2037/2001 of *Lecture Notes in Computer Science (LNCS)*, pages 441–451, Lake Como, Milan, Italy, April 18–20, 2001. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/3-540-45365-2_46.
39. Peter Salamon, Paolo Sibani, and Richard Frost. *Facts, Conjectures, and Improvements for Simulated Annealing*, volume 7 of *SIAM Monographs on Mathematical Modeling and Computation*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics (SIAM), 2002. ISBN 0898715083 and 9780898715088. URL `http://books.google.de/books?id=jhAldlYvClcC`.

40. Peter J. M. van Laarhoven and Emile H. L. Aarts, editors. *Simulated Annealing: Theory and Applications*, volume 37 of *Mathematics and its Applications*. Norwell, MA, USA: Kluwer Academic Publishers, 1987. ISBN 90-277-2513-6, 978-90-277-2513-4, and 978-90-481-8438-5. URL `http://books.google.de/books?id=-IgUab6Dp_IC`.
41. Lawrence Davis, editor. *Genetic Algorithms and Simulated Annealing*. Research Notes in Artificial Intelligence. London, UK: Pitman, 1987. ISBN 0273087711, 0934613443, 9780273087717, and 978-0934613446. URL `http://books.google.de/books?id=edfSSAAACAAJ`.
42. James C. Spall. *Introduction to Stochastic Search and Optimization*. Estimation, Simulation, and Control – Wiley-Interscience Series in Discrete Mathematics and Optimization. Chichester, West Sussex, UK: Wiley Interscience, first edition, June 2003. ISBN 0-471-33052-3, 0-471-72213-8, 978-0-471-33052-3, and 978-0-471-72213-7. URL `http://books.google.de/books?id=f66OIvvkKnAC`.
43. Scott Kirkpatrick, Charles Daniel Gelatt, Jr., and Mario P. Vecchi. Optimization by simulated annealing. *Science Magazine*, 220(4598):671–680, May 13, 1983. doi: 10.1126/science.220.4598.671. URL `http://fezzik.ucd.ie/msc/cscs/ga/kirkpatrick83optimization.pdf`.
44. Vladimír Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, January 1985. doi: 10.1007/BF00940812. URL `http://mkweb.bcgsc.ca/papers/cerny-travelingsalesman.pdf`. Communicated by S. E. Dreyfus. Also: Technical Report, Comenius University, Mlynská Dolina, Bratislava, Czechoslovakia, 1982.
45. Dean Jacobs, Jan Prins, Peter Siegel, and Kenneth Wilson. Monte carlo techniques in code optimization. *ACM SIGMICRO Newsletter*, 13(4):143–148, December 1982.
46. Dean Jacobs, Jan Prins, Peter Siegel, and Kenneth Wilson. Monte carlo techniques in code optimization. In *International Symposium on Microarchitecture – Proceedings of the 15th Annual Workshop on Microprogramming (MICRO 15)*, pages 143–146, Palo Alto, CA, USA, October 5–7, 1982. Piscataway, NJ, USA: IEEE (Institute of Electrical and Electronics Engineers).
47. Martin Pincus. A monte carlo method for the approximate solution of certain types of constrained optimization problems. *Operations Research (Oper. Res.)*, 18(6):1225–1228, November–December 1970.
48. Fred W. Glover. Tabu search – part i. *ORSA Journal on Computing*, 1(3):190–206, Summer 1989. doi: 10.1287/ijoc.1.3.190. URL `http://leeds-faculty.colorado.edu/glover/TS%20-%20Part%20I-ORSA.pdf`.
49. Fred W. Glover. Tabu search – part ii. *ORSA Journal on Computing*, 2(1):190–206, Winter 1990. doi: 10.1287/ijoc.2.1.4. URL `http://leeds-faculty.colorado.edu/glover/TS%20-%20Part%20II-ORSA-aw.pdf`.

50. Fred W. Glover and Manuel Laguna. Tabu search. In Colin R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, Advanced Topics in Computer Science Series. Chichester, West Sussex, UK: Blackwell Publishing Ltd, April 1993. ISBN 079239965X and 978-0470220795. URL http://www.dei.unipd.it/~fisch/ricop/tabu_search_glover_laguna.pdf.
51. Dominique de Werra and Alain Hertz. Tabu search techniques: A tutorial and an application to neural networks. *OR Spectrum – Quantitative Approaches in Management*, 11(3):131–141, September 1989. doi: 10.1007/BF01720782. URL http://www.springerlink.de/content/x25k97k0qx237553/fulltext.pdf.
52. Roberto Battiti and Giampietro Tecchiolli. The reactive tabu search. *ORSA Journal on Computing*, 6(2):126–140, 1994. doi: 10.1287/ijoc.6.2.126. URL http://citeseer.ist.psu.edu/141556.html.
53. 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.
54. 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.
55. 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.
56. 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.
57. 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.
58. 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.

59. 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.
60. 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.
61. 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.
62. Gregory Z. Gutin and Abraham P. Punnen, editors. *The Traveling Salesman Problem and its Variations*, volume 12 of *Combinatorial Optimization*. Norwell, MA, USA: Kluwer Academic Publishers, 2002. ISBN 0-306-48213-4, 1-4020-0664-0, and 978-1-4020-0664-7. doi: 10.1007/b101971. URL http://books.google.de/books?id=TRYkPg_Xf20C.
63. Weiqi Li. Seeking global edges for traveling salesman problem in multi-start search. *Journal of Global Optimization*, 51(3): 515–540, November 2011. doi: 10.1007/s10898-010-9643-4.
64. Sami Khuri, Martin Schütz, and Jörg Heitkötter. Evolutionary heuristics for the bin packing problem. In David W. Pearson, Nigel C. Steele, and Rudolf F. Albrecht, editors, *Proceedings of the 2nd International Conference on Artificial Neural Nets and Genetic Algorithms (ICANNGA'95)*, pages 285–288, Alès, France, April 18–21, 1995. New York, NY, USA: Springer New York. URL http://www6.uniovi.es/pub/EC/GA/papers/icannga95.ps.gz.
65. Holger H. Hoos and Thomas Stützle. Satlib: An online resource for research on sat. In Ian Gent, Hans van Maaren, and Toby Walsh, editors, *SAT2000 – Highlights of Satisfiability Research in the Year 2000*, volume 63 of *Frontiers in Artificial Intelligence and Applications*, pages 283–292. Amsterdam, The Netherlands: IOS Press, 2000. URL http://www.cs.ubc.ca/~hoos/Publ/sat2000-satlib.pdf.

66. Dave Andrew Douglas Tompkins and Holger H. Hoos. Ubcsat: An implementation and experimentation environment for sls algorithms for sat and max-sat. In Holger H. Hoos and David G. Mitchell, editors, *Revised Selected Papers from the Seventh International Conference on Theory and Applications of Satisfiability Testing (SAT'04)*, volume 3542 of *Lecture Notes in Computer Science (LNCS)*, pages 306–320, Vancouver, BC, Canada, May 10–13, 2004. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/11527695_24. URL http://ubcsat.dtompkins.com/downloads/sat04proc-ubcsat.pdf.
67. Thomas J. Schaefer. The complexity of satisfiability problems. In Richard J. Lipton, Walter Burkhard, Walter Savitch, Emily P. Friedman, and Alfred Vaino Aho, editors, *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing (STOC'78)*, pages 216–226, San Diego, CA, USA, May 1–3, 1978. New York, NY, USA: Association for Computing Machinery (ACM). doi: 10.1145/800133.804350. URL http://www.ccs.neu.edu/home/lieber/courses/csg260/f06/materials/papers/max-sat/p216-schaefer.pdf.
68. Claudio Rossi, Elena Marchiori, and Joost N. Kok. An adaptive evolutionary algorithm for the satisfiability problem. In *Proceedings of the 2000 ACM symposium on Applied computing (SAC'00)*, volume 1, pages 463–469, Villa Olmo, Como, Italy, March 19–21, 2000. New York, NY, USA: ACM Press. doi: 10.1145/335603.335912. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.4771.
69. Peter Merz and Bernd Freisleben. A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem. In Peter John Angeline, Zbigniew Michalewicz, Marc Schoenauer, Xin Yao, and Ali M. S. Zalzala, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'99)*, volume 3, pages 2063–2070, Washington, DC, USA: Mayflower Hotel, July 6–9, 1999. Piscataway, NJ, USA: IEEE Computer Society. URL http://en.scientificcommons.org/204950.
70. Luca Maria Gambardella, Éric D. Taillard, and Marco Dorigo. Ant colonies for the quadratic assignment problem. *The Journal of the Operational Research Society (JORS)*, 50(2):167–176, February 1999. doi: 10.2307/3010565. URL http://www.idsia.ch/~luca/tr-idsia-4-97.pdf.
71. Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros. Real-parameter black-box optimization benchmarking 2010: Experimental setup. Rapports de Recherche 7215, Institut National de Recherche en Informatique et en Automatique (INRIA), March 9, 2010. URL http://hal.inria.fr/docs/00/46/24/81/PDF/RR-7215.pdf.
72. Thomas Weise, Raymond Chiong, Ke Tang, Jörg Lässig, Shigeyoshi Tsutsui, Wenxiang Chen, Zbigniew Michalewicz, and Xin Yao. Benchmarking optimization algorithms: An open source framework for the traveling salesman problem. *IEEE Computational Intelligence Magazine (CIM)*, 9(3):40–52, August 2014. doi: 10.1109/MCI.2014.2326101. URL http://www.it-weise.de/documents/files/WCTLTCMY2014BOAAOSFFTTSP.pdf. Featured article and selected paper at the website of the IEEE Computational Intelligence Society (http://cis.ieee.org/).

73. Mark S. Boddy and Thomas L. Dean. Solving time-dependent planning problems. Technical Report CS-89-03, Providence, RI, USA: Brown University, Department of Computer Science, February 1989. URL ftp://ftp.cs.brown.edu/pub/techreports/89/cs89-03.pdf.
74. John D. C. Little, Katta G. Murty, Dura W. Sweeny, and Caroline Karel. An algorithm for the traveling salesman problem. Sloan Working Papers 07-63, Cambridge, MA, USA: Massachusetts Institute of Technology (MIT), Sloan School of Management, March 1, 1963. URL http://dspace.mit.edu/bitstream/handle/1721.1/46828/algorithmfortrav00litt.pdf.
75. Weixiong Zhang. Truncated branch-and-bound: A case study on the asymmetric traveling salesman problem. In *Proceedings of the AAAI-93 Spring Symposium on AI and NP-Hard Problems*, pages 160–166, Stanford, CA, USA, March 23–25, 1993. Menlo Park, CA, USA: AAAI Press. URL www.cs.wustl.edu/~zhang/publications/atsp-aaai93-symp.ps.
76. Weixiong Zhang. Truncated and anytime depth-first branch and bound: A case study on the asymmetric traveling salesman problem. In Weixiong Zhang and Sven König, editors, *AAAI Spring Symposium Series: Search Techniques for Problem Solving Under Uncertainty and Incomplete Information*, volume SS-99-07 of *AAAI Technical Report*, pages 148–155. Menlo Park, CA, USA: AAAI Press, March 1999. URL https://www.aaai.org/Papers/Symposia/Spring/1999/SS-99-07/SS99-07-026.pdf.