Metaheuristics for Smart Manufacturing

9. Why is optimization difficult?

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

1. Premature Convergence
2. Ruggedness & Causality
3. Deceptiveness & Neutrality
4. Epistasis
5. Scalability
6. Summary

The slides are available at http://iao.hfuu.edu.cn/155, the book at http://thomasweise.github.io/aitoa, and the source code at http://www.github.com/thomasWeise/aitoa-code

course book

course material

An Introduction to Optimization Algorithms

The contents of this course are available as free electronic book *"An Introduction to Optimization Algorithms"* [1] at http://thomasweise.github.io/aitoa in pdf, html, azw3, and epub format, created with our bookbuildeR tool chain.

Why is optimization difficult?

- So far, we have discussed several different optimization methods.

Why is optimization difficult?

- So far, we have discussed several different optimization methods.
- You now have some experience in solving optimization problems.

Why is optimization difficult?

- So far, we have discussed several different optimization methods.
- You now have some experience in solving optimization problems.
- In optimization, there are problems that tend to be easy, where we can get good solutions quickly.

Why is optimization difficult?

- So far, we have discussed several different optimization methods.
- You now have some experience in solving optimization problems.
- In optimization, there are problems that tend to be easy, where we can get good solutions quickly.
- There are also other problems, where it takes a long time to even gut some barely acceptable results.

- So far, we have discussed several different optimization methods.
- You now have some experience in solving optimization problems.
- In optimization, there are problems that tend to be easy, where we can get good solutions quickly.
- There are also other problems, where it takes a long time to even gut some barely acceptable results.
- Why?

- So far, we have discussed several different optimization methods.
- You now have some experience in solving optimization problems.
- In optimization, there are problems that tend to be easy, where we can get good solutions quickly.
- There are also other problems, where it takes a long time to even gut some barely acceptable results.
- Why?
- What makes optimization difficult?

Why is optimization difficult?

- Some things are pretty clear

Why is optimization difficult?

- Some things are pretty clear:
  - For some problems (e.g., the finding shortest paths in a graph) there are deterministic algorithms that can give the optimal solutions quickly.

Why is optimization difficult?

- Some things are pretty clear:
  - For some problems (e.g., the finding shortest paths in a graph) there are deterministic algorithms that can give the optimal solutions quickly.
  - For other problems, either no such method exists or it is not feasible (too slow).

- Some things are pretty clear:
  - For some problems (e.g., the finding shortest paths in a graph) there are deterministic algorithms that can give the optimal solutions quickly.
  - For other problems, either no such method exists or it is not feasible (too slow).
  - We have looked on some of these problems (Job Shop Scheduling, TSP, ...)

- Some things are pretty clear:
  - For some problems (e.g., the finding shortest paths in a graph) there are deterministic algorithms that can give the optimal solutions quickly.
  - For other problems, either no such method exists or it is not feasible (too slow).
  - We have looked on some of these problems (Job Shop Scheduling, TSP, ...)
  - And used different algorithms to solve the Job Shop Scheduling Problem

- Some things are pretty clear:
  - For some problems (e.g., the finding shortest paths in a graph) there are deterministic algorithms that can give the optimal solutions quickly.
  - For other problems, either no such method exists or it is not feasible (too slow).
  - We have looked on some of these problems (Job Shop Scheduling, TSP, ...)
  - And used different algorithms to solve the Job Shop Scheduling Problem
  - Some algorithms worked well, some worked not so well.

- Some things are pretty clear:
  - For some problems (e.g., the finding shortest paths in a graph) there are deterministic algorithms that can give the optimal solutions quickly.
  - For other problems, either no such method exists or it is not feasible (too slow).
  - We have looked on some of these problems (Job Shop Scheduling, TSP, ...)
  - And used different algorithms to solve the Job Shop Scheduling Problem
  - Some algorithms worked well, some worked not so well.
  - The solution quality we get depends on the algorithm we use.

Why is optimization difficult?

- Some things are pretty clear:
  - For some problems (e.g., the finding shortest paths in a graph) there are deterministic algorithms that can give the optimal solutions quickly.
  - For other problems, either no such method exists or it is not feasible (too slow).
  - We have looked on some of these problems (Job Shop Scheduling, TSP, ...)
  - And used different algorithms to solve the Job Shop Scheduling Problem
  - Some algorithms worked well, some worked not so well.
  - The solution quality we get depends on the algorithm we use.
  - Different algorithms deal with the difficulties of optimization tasks differently.

Why is optimization difficult?

- Some things are pretty clear:
  - For some problems (e.g., the finding shortest paths in a graph) there are deterministic algorithms that can give the optimal solutions quickly.
  - For other problems, either no such method exists or it is not feasible (too slow).
  - We have looked on some of these problems (Job Shop Scheduling, TSP, ...)
  - And used different algorithms to solve the Job Shop Scheduling Problem
  - Some algorithms worked well, some worked not so well.
  - The solution quality we get depends on the algorithm we use.
  - Different algorithms deal with the difficulties of optimization tasks differently.
  - There may be different aspects that render a problem difficult.

Preface

- Assume we chose to use a randomized metaheuristic algorithm that gives an *approximate* solution for a problem.

Preface

- Assume we chose to use a randomized metaheuristic algorithm that gives an *approximate* solution for a problem.
- We cannot expect to get the global optimum.

- Assume we chose to use a randomized metaheuristic algorithm that gives an *approximate* solution for a problem.
- We cannot expect to get the global optimum.
- We are interested in the quality of the solutions that we can get.

Preface

- Assume we chose to use a randomized metaheuristic algorithm that gives an *approximate* solution for a problem.
- We cannot expect to get the global optimum.
- We are interested in the quality of the solutions that we can get.
- What features of problems or algorithms allow us to get good solutions?

Preface

- Assume we chose to use a randomized metaheuristic algorithm that gives an *approximate* solution for a problem.
- We cannot expect to get the global optimum.
- We are interested in the quality of the solutions that we can get.
- What features of problems or algorithms allow us to get good solutions?
- When/why can we not get good solutions?

- Let us start at the end!

Convergence

- Let us start at the end!
- Sooner or later, every optimization process ends (converges)

- Let us start at the end!
- Sooner or later, every optimization process ends (converges)

Definition (Convergence)

An optimization algorithm has converged if it cannot reach new candidate solutions anymore or if it keeps on producing candidate solutions from a small subset of the solution space $\mathbb{Y}$. [2, 3]

- Let us start at the end!
- Sooner or later, every optimization process ends (converges)

Definition (Convergence)

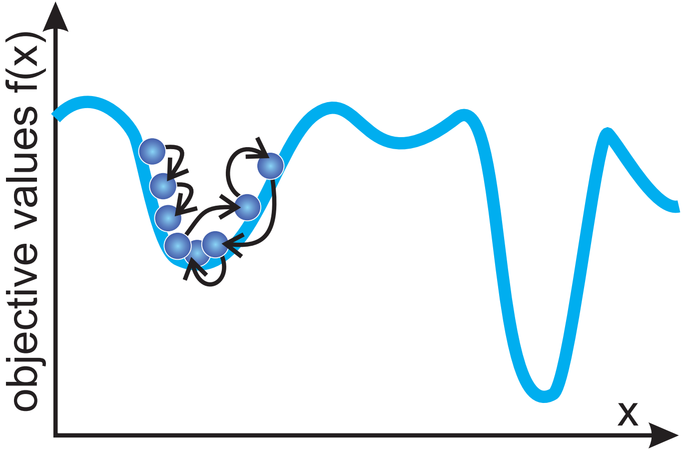
An optimization algorithm has converged if it cannot reach new candidate solutions anymore or if it keeps on producing candidate solutions from a small subset of the solution space $\mathbb{Y}$. [2, 3]

Definition (multi-modality)

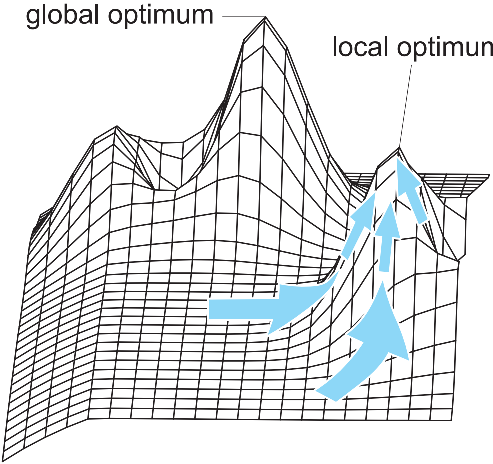
A function/optimization problem is multi-modal if it has more than one minimum / maximum /optimum. [4–8]

Section Outline

- Premature Convergence = convergence to local optimum [2, 3]



- Premature Convergence = convergence to local optimum [2, 3]



- We have two choices what to do in optimization

- We have two choices what to do in optimization:
  1. Exploration: search in distant areas of the search space, strong randomization: slow improvement

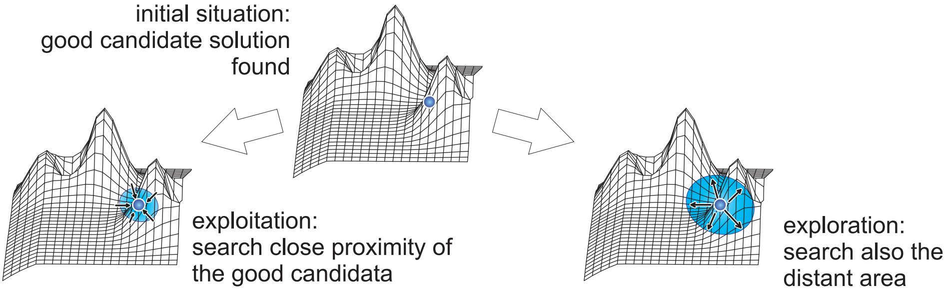
Exploration versus Exploitation

- We have two choices what to do in optimization:
  1. Exploration: search in distant areas of the search space, strong randomization: slow improvement
  2. Exploitation: analyze neighborhood of current best solutions: fast improvement/convergence

Exploration versus Exploitation

- We have two choices what to do in optimization:
  1. Exploration: search in distant areas of the search space, strong randomization: slow improvement
  2. Exploitation: analyze neighborhood of current best solutions: fast improvement/convergence
- Which should we use more?

- Which should we use more?
- This is called the Exploration versus Exploitation Dilemma [9–16]



Countermeasures

- Countermeasures against Premature Convergence
  1. Restarting [17, 18]

Countermeasures

- Countermeasures against Premature Convergence
  1. Restarting [17, 18]:
     - if no improvement for some time, restart algorithm

Countermeasures

- Countermeasures against Premature Convergence
  1. Restarting [17, 18]:
     - if no improvement for some time, restart algorithm
     - in case of EA: restart with bigger population size

Countermeasures

- Countermeasures against Premature Convergence
  1. Restarting [17, 18]:
     - if no improvement for some time, restart algorithm
     - in case of EA: restart with bigger population size
  2. Low selection pressure and/or larger population size [19–21]

Countermeasures

- Countermeasures against Premature Convergence
  1. Restarting [17, 18]:
     - if no improvement for some time, restart algorithm
     - in case of EA: restart with bigger population size
  2. Low selection pressure and/or larger population size [19–21]:
     - allows for more exploration by putting less pressure to move to better solutions
     - slows down search

Countermeasures

- Countermeasures against Premature Convergence
  1. Restarting [17, 18]:
     - if no improvement for some time, restart algorithm
     - in case of EA: restart with bigger population size
  2. Low selection pressure and/or larger population size [19–21]:
     - allows for more exploration by putting less pressure to move to better solutions
     - slows down search
     - only sometimes helpful [21, 22]

Countermeasures

- Countermeasures against Premature Convergence
  1. Restarting [17, 18]:
     - if no improvement for some time, restart algorithm
     - in case of EA: restart with bigger population size
  2. Low selection pressure and/or larger population size [19–21]:
     - allows for more exploration by putting less pressure to move to better solutions
     - slows down search
     - only sometimes helpful [21, 22]
  3. Sometimes accepting worse candidate solutions, as in, e.g., Simulated Annealing [23–26]

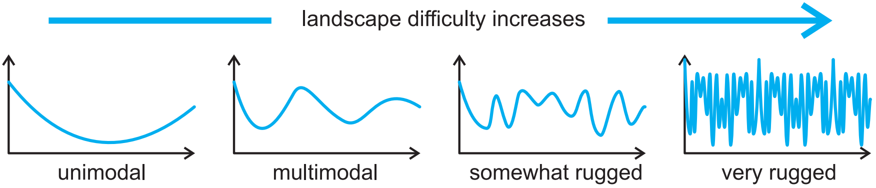
Countermeasures

- Countermeasures against Premature Convergence
  1. Restarting [17, 18]:
     - if no improvement for some time, restart algorithm
     - in case of EA: restart with bigger population size
  2. Low selection pressure and/or larger population size [19–21]:
     - allows for more exploration by putting less pressure to move to better solutions
     - slows down search
     - only sometimes helpful [21, 22]
  3. Sometimes accepting worse candidate solutions, as in, e.g., Simulated Annealing [23–26]
  4. Sharing, Niching, and Clearing in EAs [27–40]

Countermeasures

- Countermeasures against Premature Convergence
  1. Restarting [17, 18]:
     - if no improvement for some time, restart algorithm
     - in case of EA: restart with bigger population size
  2. Low selection pressure and/or larger population size [19–21]:
     - allows for more exploration by putting less pressure to move to better solutions
     - slows down search
     - only sometimes helpful [21, 22]
  3. Sometimes accepting worse candidate solutions, as in, e.g., Simulated Annealing [23–26]
  4. Sharing, Niching, and Clearing in EAs [27–40]:
     - In an EA, give solutions that are very similar to each other a worse fitness

Ruggedness and Weak Causality



- Why??

- Basic assumption behind metaheuristic optimization:

- Basic assumption behind metaheuristic optimization:

Definition (Strong Causality)

Small changes to an object should lead to small changes in its behavior / objective values. [41–43]

Causality

- Basic assumption behind metaheuristic optimization:

Definition (Strong Causality)

Small changes to an object should lead to small changes in its behavior / objective values. [41–43]

- What happens if the causality in a problem is weak?

- Basic assumption behind metaheuristic optimization:

Definition (Strong Causality)

Small changes to an object should lead to small changes in its behavior / objective values. [41–43]

- What happens if the causality in a problem is weak?
- The "memory" inside the search becomes useless.

- Basic assumption behind metaheuristic optimization:

Definition (Strong Causality)

Small changes to an object should lead to small changes in its behavior / objective values. [41–43]

- What happens if the causality in a problem is weak?
- The "memory" inside the search becomes useless, because a better solution may appear anywhere, not just next to a good solution (and the same holds for worse solutions the other way around...)

Countermeasures

1. Hybridization of EAs with local search:

Countermeasures

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]

Countermeasures

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]:
     - EA + local search in the search space

Countermeasures

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]:
     - EA + local search in the search space
     - Each point in the search space is generated by the normal search operations and then refined with a local search

Countermeasures

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]:
     - EA + local search in the search space
     - Each point in the search space is generated by the normal search operations and then refined with a local search
     - The fitness landscape appears smooth to the EA, as it only sees local optima and not the rugged spikes in between

Countermeasures

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]
   - Baldwin effect [45–48]

Countermeasures

1. Hybridization of EAs with local search:
    - Lamarckian evolution [44, 45]
    - Baldwin effect [45–48]:
        - EA + local search in the solution space

Countermeasures

1. Hybridization of EAs with local search:
    - Lamarckian evolution [44, 45]
    - Baldwin effect [45–48]:
        - EA + local search in the solution space
        - Each candidate solution is the result of the mapping γ applied to a point in the search süace

Countermeasures

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]
   - Baldwin effect [45–48]:
     - EA + local search in the solution space
     - Each candidate solution is the result of the mapping γ applied to a point in the search süace
     - The candidate solution is then refined with a local search

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]
   - Baldwin effect [45–48]:
     - EA + local search in the solution space
     - Each candidate solution is the result of the mapping γ applied to a point in the search süace
     - The candidate solution is then refined with a local search
     - The fitness landscape appears smooth to the EA, as it only sees local optima and not the rugged spikes in between

Countermeasures

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]
   - Baldwin effect [45–48]
   - Memetic Algorithms [49–58]

Countermeasures

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]
   - Baldwin effect [45–48]
   - Memetic Algorithms [49–58]:
     - Similar to Baldwin effect and Lamarckian evolution

Countermeasures

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]
   - Baldwin effect [45–48]
   - Memetic Algorithms [49–58]:
     - Similar to Baldwin effect and Lamarckian evolution
     - The search operations themselves perform some local optimization

Countermeasures

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]
   - Baldwin effect [45–48]
   - Memetic Algorithms [49–58]:
     - Similar to Baldwin effect and Lamarckian evolution
     - The search operations themselves perform some local optimization
     - The fitness landscape appears smooth to the EA, as it only sees local optima and not the rugged spikes in between

Countermeasures

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]
   - Baldwin effect [45–48]
   - Memetic Algorithms [49–58]
   - other hybrid approaches [59–66]

Countermeasures

1. Hybridization of EAs with local search:
    - Lamarckian evolution [44, 45]
    - Baldwin effect [45–48]
    - Memetic Algorithms [49–58]
    - other hybrid approaches [59–66]:
        - Combinations of Evolutionary Algorithms with local search, or combinations of Evolutionary Algorithms with other concepts from Machine Learning

Countermeasures

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]
   - Baldwin effect [45–48]
   - Memetic Algorithms [49–58]
   - other hybrid approaches [59–66]
2. Landscape approximation [67]

Countermeasures

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]
   - Baldwin effect [45–48]
   - Memetic Algorithms [49–58]
   - other hybrid approaches [59–66]
2. Landscape approximation [67]:
   - Try to adjust the parameters of a (simple) model M or function so that it behaves similar to the (points so-far seen from the) objective function f

Countermeasures

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]
   - Baldwin effect [45–48]
   - Memetic Algorithms [49–58]
   - other hybrid approaches [59–66]
2. Landscape approximation [67]:
   - Try to adjust the parameters of a (simple) model M or function so that it behaves similar to the (points so-far seen from the) objective function f
   - Optimize on this simple model (which has stronger causality)

Countermeasures

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]
   - Baldwin effect [45–48]
   - Memetic Algorithms [49–58]
   - other hybrid approaches [59–66]
2. Landscape approximation [67]:
   - Try to adjust the parameters of a (simple) model M or function so that it behaves similar to the (points so-far seen from the) objective function f
   - Optimize on this simple model (which has stronger causality)
   - After a few steps, go back to original f and test solutions

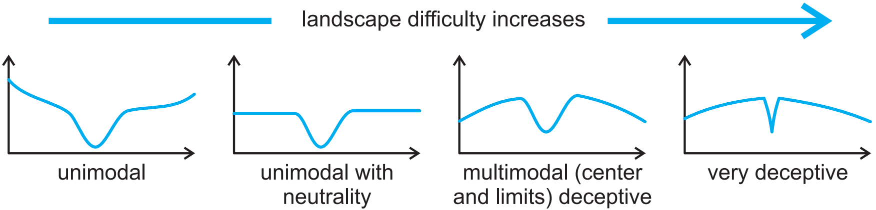
Countermeasures

1. Hybridization of EAs with local search:
   - Lamarckian evolution [44, 45]
   - Baldwin effect [45–48]
   - Memetic Algorithms [49–58]
   - other hybrid approaches [59–66]
2. Landscape approximation [67]:
   - Try to adjust the parameters of a (simple) model M or function so that it behaves similar to the (points so-far seen from the) objective function f
   - Optimize on this simple model (which has stronger causality)
   - After a few steps, go back to original f and test solutions
   - Update M, then repeat

Section Outline

Deceptiveness

- Gradient and information lead optimizer away from optimum [60, 68, 69]



- Why??

Countermeasures

- Countermeasures

Countermeasures

- Countermeasures
  1. Choose appropriate representation, maybe combine representations [70]

- Countermeasures
  1. Choose appropriate representation, maybe combine representations [70]
  2. Preventing convergence:

- Countermeasures
  1. Choose appropriate representation, maybe combine representations [70]
  2. Preventing convergence:
     - Fitness Uniform Selection Scheme [71–74]

Countermeasures

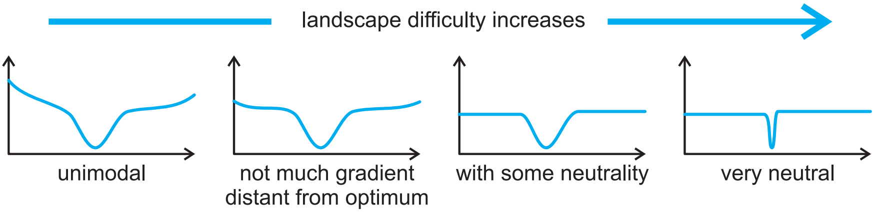
- Countermeasures
  1. Choose appropriate representation, maybe combine representations [70]
  2. Preventing convergence:
     - Fitness Uniform Selection Scheme [71–74]
     - Novelty Search [75–77]

Countermeasures

- Countermeasures
  1. Choose appropriate representation, maybe combine representations [70]
  2. Preventing convergence:
     - Fitness Uniform Selection Scheme [71–74]
     - Novelty Search [75–77]
     - Frequency Fitness Assignment [78]

Neutrality

- Neutrality: Many candidate solutions have same objective values
- Little or no information gained from sampling the solution space



- Why??
- Countermeasures: Same as for Deceptiveness

Section Outline

1. Premature Convergence
2. Ruggedness & Causality
3. Deceptiveness & Neutrality
4. Epistasis
5. Scalability
6. Summary

Epistasis is a concept from biology, which we can similarly find in optimization. . .

Epistasis is a concept from biology, which we can similarly find in optimization. . .

Definition (Epistasis)

One gene influences the behavior (contribution to the objective function) of other genes [34, 79–86]

Epistasis is a concept from biology, which we can similarly find in optimization. . .

Definition (Epistasis)

One gene influences the behavior (contribution to the objective function) of other genes [34, 79–86]

Definition (Separability)

A function of n variables is separable if it can be rewritten as a sum of n functions of just one variable [87–90]

Epistasis: The other root of all evil...

Epistasis is a concept from biology, which we can similarly find in optimization...

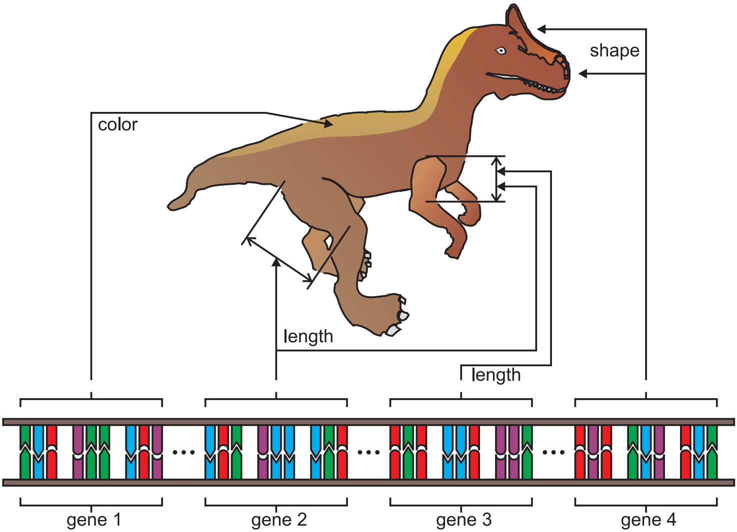
Definition (Epistasis)

One gene influences the behavior (contribution to the objective function) of other genes [34, 79–86]

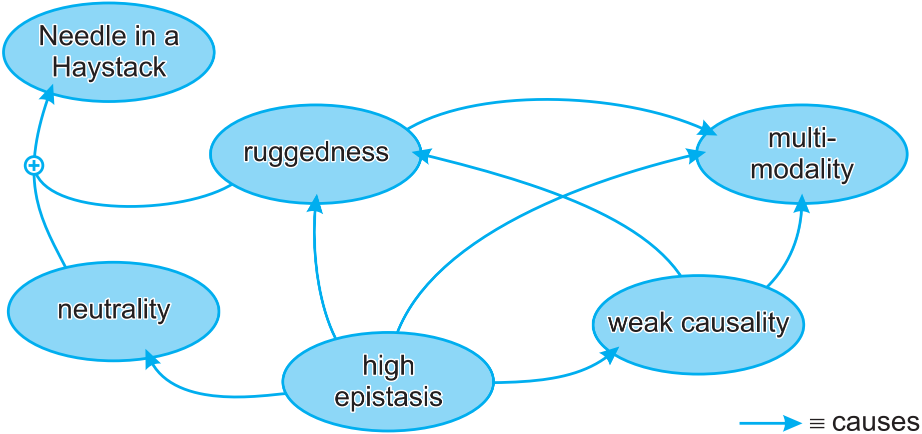
Definition (Separability)

A function of n variables is separable if it can be rewritten as a sum of n functions of just one variable [87–90]

- non-epistatic (separable) problems can be solved efficiently by decomposition


shape
color
length
length
gene 1
gene 2
gene 3
gene 4

Influence of Epistasis



Countermeasures

1. See countermeasures for ruggedness, deceptiveness, multi-modality. . .

Countermeasures

1. See countermeasures for ruggedness, deceptiveness, multi-modality. . .
2. Choose appropriate representation [91–94] and search operators [95]

Countermeasures

1. See countermeasures for ruggedness, deceptiveness, multi-modality...
2. Choose appropriate representation [91–94] and search operators [95]
3. Parameter Tweaking [95]

Countermeasures

1. See countermeasures for ruggedness, deceptiveness, multi-modality. . .
2. Choose appropriate representation [91–94] and search operators [95]
3. Parameter Tweaking [95]
4. Linkage learning [96–102] and Variable Interaction Learning [103]

Countermeasures

1. See countermeasures for ruggedness, deceptiveness, multi-modality...
2. Choose appropriate representation [91–94] and search operators [95]
3. Parameter Tweaking [95]
4. Linkage learning [96–102] and Variable Interaction Learning [103]:
    - Try to find out which variables in the points of the search space are (epistatically) linked together

Countermeasures

1. See countermeasures for ruggedness, deceptiveness, multi-modality...
2. Choose appropriate representation [91–94] and search operators [95]
3. Parameter Tweaking [95]
4. Linkage learning [96–102] and Variable Interaction Learning [103]:
   - Try to find out which variables in the points of the search space are (epistatically) linked together
   - Try to change these variables only together, consider them as a unit

Countermeasures

1. See countermeasures for ruggedness, deceptiveness, multi-modality...
2. Choose appropriate representation [91–94] and search operators [95]
3. Parameter Tweaking [95]
4. Linkage learning [96–102] and Variable Interaction Learning [103]:
   - Try to find out which variables in the points of the search space are (epistatically) linked together
   - Try to change these variables only together, consider them as a unit
   - For example: In binary operator, try to always pass such variables together to the offspring and to not separate them

Countermeasures

1. See countermeasures for ruggedness, deceptiveness, multi-modality…
2. Choose appropriate representation [91–94] and search operators [95]
3. Parameter Tweaking [95]
4. Linkage learning [96–102] and Variable Interaction Learning [103]
5. If epistasis is limited: cooperative-coevolution approach [103–105] (see later)

Section Outline

Scalability

- Time required for solving $\mathcal{NP}$-hard problems grows exponential with input size

- Time required for solving $\mathcal{NP}$-hard problems grows exponential with input size
- Metaheuristic optimization: approximately solve $\mathcal{NP}$-hard problems in feasible time

Scalability

- Time required for solving $\mathcal{NP}$-hard problems grows exponential with input size
- Metaheuristic optimization: approximately solve $\mathcal{NP}$-hard problems in feasible time
- ...but their time requirement also grows with problem size...

Scalability

- Time required for solving $\mathcal{NP}$-hard problems grows exponential with input size
- Metaheuristic optimization: approximately solve $\mathcal{NP}$-hard problems in feasible time
- ... but their time requirement also grows with problem size...
- "Curse of Dimensionality": solution space volume increases exponentially with number of decision variables (genes) [106, 107]

Scalability

- Time required for solving $\mathcal{NP}$-hard problems grows exponential with input size
- Metaheuristic optimization: approximately solve $\mathcal{NP}$-hard problems in feasible time
- ...but their time requirement also grows with problem size...
- "Curse of Dimensionality": solution space volume increases exponentially with number of decision variables (genes) [106, 107]
- Example: search in $(1 \ldots 10)^n$

Scalability

- Time required for solving $\mathcal{NP}$-hard problems grows exponential with input size
- Metaheuristic optimization: approximately solve $\mathcal{NP}$-hard problems in feasible time
- ... but their time requirement also grows with problem size...
- "Curse of Dimensionality": solution space volume increases exponentially with number of decision variables (genes) [106, 107]
- Example: search in $(1 \dots 10)^n$
- any algorithm (for non-trivial problems) takes longer for larger inputs...

Countermeasures

- Countermeasures

- Countermeasures
  1. Parallelization and distribution

- Countermeasures
  1. Parallelization and distribution:
     - sub-linear speed-up can be achieved [116]

Countermeasures

- Countermeasures
  1. Parallelization and distribution:
     - sub-linear speed-up can be achieved [116]
     - Parallelization: Use multi-core CPU + multiple threads

- Countermeasures
  1. Parallelization and distribution:
     - sub-linear speed-up can be achieved [116]
     - Parallelization: Use multi-core CPU + multiple threads or GPUs [117–122]

Countermeasures

- Countermeasures
  1. Parallelization and distribution:
     - sub-linear speed-up can be achieved [116]
     - Parallelization: Use multi-core CPU + multiple threads or GPUs [117–122]
     - Distribution: Use multiple computers in a network [123, 124], a cluster, or a grid

- Countermeasures
  1. Parallelization and distribution
  2. Indirect Representation 1: Generative [108, 109]

- Countermeasures
  1. Parallelization and distribution
  2. Indirect Representation 1: Generative [108, 109]:
     - points in search space have few variables, search space is smaller, can be explored more easily

Countermeasures

- Countermeasures
  1. Parallelization and distribution
  2. Indirect Representation 1: Generative [108, 109]:
     - points in search space have few variables, search space is smaller, can be explored more easily
     - They are mapped to larger, more complex candidate solutions by a simple functional mapping γ

- Countermeasures
  1. Parallelization and distribution
  2. Indirect Representation 1: Generative [108, 109]:
     - points in search space have few variables, search space is smaller, can be explored more easily
     - They are mapped to larger, more complex candidate solutions by a simple functional mapping γ
     - Utilizes/assumes symmetries in the candidate solutions

Countermeasures

- Countermeasures
  1. Parallelization and distribution
  2. Indirect Representation 1: Generative [108, 109]
  3. Indirect Representation 2: Development [110, 111]

Countermeasures

- Countermeasures
  1. Parallelization and distribution
  2. Indirect Representation 1: Generative [108, 109]
  3. Indirect Representation 2: Development [110, 111]:
     - Similar to generative mapping, the search space is smaller

- Countermeasures
  1. Parallelization and distribution
  2. Indirect Representation 1: Generative [108, 109]
  3. Indirect Representation 2: Development [110, 111]:
     - Similar to generative mapping, the search space is smaller
     - But: Mapping γ is more complex, a simulation which incorporates feedback from an environment or the objective function

Countermeasures

- Countermeasures
  1. Parallelization and distribution
  2. Indirect Representation 1: Generative [108, 109]
  3. Indirect Representation 2: Development [110, 111]:
     - Similar to generative mapping, the search space is smaller
     - But: Mapping γ is more complex, a simulation which incorporates feedback from an environment or the objective function
     - Better behavior than generative mappings

Countermeasures

- Countermeasures
  1. Parallelization and distribution
  2. Indirect Representation 1: Generative [108, 109]
  3. Indirect Representation 2: Development [110, 111]
  4. Exploiting Separability, e.g., with coevolution [103–105, 112, 113]

Countermeasures

- Countermeasures
  1. Parallelization and distribution
  2. Indirect Representation 1: Generative [108, 109]
  3. Indirect Representation 2: Development [110, 111]
  4. Exploiting Separability, e.g., with coevolution [103–105, 112, 113]:
     - Try to divide the problem into (almost) unrelated problems with smaller search spaces

Countermeasures

- Countermeasures
  1. Parallelization and distribution
  2. Indirect Representation 1: Generative [108, 109]
  3. Indirect Representation 2: Development [110, 111]
  4. Exploiting Separability, e.g., with coevolution [103–105, 112, 113]:
     - Try to divide the problem into (almost) unrelated problems with smaller search spaces
     - Solve them more or less separately, combine solutions to get overall solution, and repeat

- Countermeasures
  1. Parallelization and distribution
  2. Indirect Representation 1: Generative [108, 109]
  3. Indirect Representation 2: Development [110, 111]
  4. Exploiting Separability, e.g., with coevolution [103–105, 112, 113]:
     - Try to divide the problem into (almost) unrelated problems with smaller search spaces
     - Solve them more or less separately, combine solutions to get overall solution, and repeat
     - Cooperative Coevolution [103, 105]: Use an EA that can find our how to divide the problem by itself and then applies the above

Countermeasures

- Countermeasures
  1. Parallelization and distribution
  2. Indirect Representation 1: Generative [108, 109]
  3. Indirect Representation 2: Development [110, 111]
  4. Exploiting Separability, e.g., with coevolution [103–105, 112, 113]
  5. Using multiple algorithms at once [114] or portfolios [115]

Section Outline

- Question: Can an optimization algorithm A be better than algorithm B?

No Free Lunch Theorem

- Question: Can an optimization algorithm A be better than algorithm B?
- Question: Can an optimization algorithm A be better than Random Sampling?

No Free Lunch Theorem

- Question: Can an optimization algorithm A be better than algorithm B?
- Question: Can an optimization algorithm A be better than Random Sampling?
- Wolpert and Macready [125] – No Free Lunch Theorem: Over all optimization problems ϕ over finite domains, the sum of the probabilities to reach a certain objective value y after m steps with algorithm A is the same as with algorithm B

$$\sum_{\forall\phi} P(y|\phi, m, A) = \sum_{\forall\phi} P(y|\phi, m, B) \tag{1}$$

No Free Lunch Theorem

- According to the No Free Lunch Theorem, the answers are:

No Free Lunch Theorem

- According to the No Free Lunch Theorem, the answers are:
- Question: Can an optimization algorithm A be better than algorithm B?

No Free Lunch Theorem

- According to the No Free Lunch Theorem, the answers are:
- Question: Can an optimization algorithm A be better than algorithm B? Not for all problems!

No Free Lunch Theorem

- According to the No Free Lunch Theorem, the answers are:
- Question: Can an optimization algorithm A be better than algorithm B? Not for all problems!
- Question: Can an optimization algorithm A be better than a Random Walk?

No Free Lunch Theorem

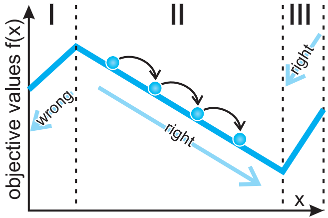
- According to the No Free Lunch Theorem, the answers are:
- Question: Can an optimization algorithm A be better than algorithm B? Not for all problems!
- Question: Can an optimization algorithm A be better than a Random Walk? Not for all problems!

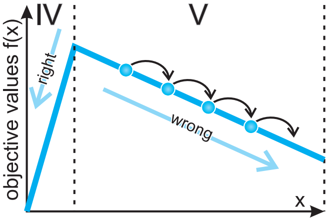
No Free Lunch Theorem

- According to the No Free Lunch Theorem, the answers are:
- Question: Can an optimization algorithm A be better than algorithm B? Not for all problems!
- Question: Can an optimization algorithm A be better than a Random Walk? Not for all problems!
- Put simply: For every problem where method A works well, we can construct a problem where the method does not work

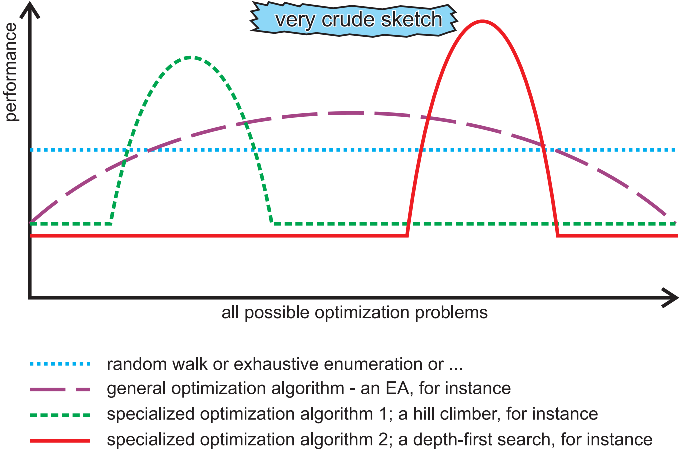
No Free Lunch Theorem

- According to the No Free Lunch Theorem, the answers are:
- Question: Can an optimization algorithm A be better than algorithm B? Not for all problems!
- Question: Can an optimization algorithm A be better than a Random Walk? Not for all problems!
- Put simply: For every problem where method A works well, we can construct a problem where the method does not work

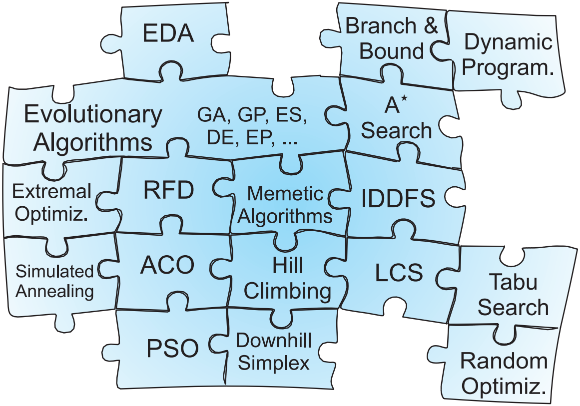




- Different algorithms are good for different problems



- Different algorithms are good for different problems and not all possible problems actually occur in practice [34, 126, 127]



Summary

- Optimization is difficult
- Metaheuristic optimizers may converge prematurely or non-uniformly
- Ruggedness is not good
- Deceptiveness is not good
- Neutrality is not good
- Epistasis is always bad – and often a representation issue!
- Large problem scales are not good
- No Free Lunch Theorem

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. Thomas Weise. *An Introduction to Optimization Algorithms*. Institute of Applied Optimization (IAO), Faculty of Computer Science and Technology, Hefei University, Hefei, Anhui, China, 2019-06-25 edition, 2018–2019. URL http://thomasweise.github.io/aitoa/. see also [34].
2. Rasmus K. Ursem. *Models for Evolutionary Algorithms and Their Applications in System Identification and Control Optimization*. PhD thesis, University of Aarhus, Department of Computer Science, Århus, Denmark, April 2003. URL http://www.daimi.au.dk/~ursem/publications/RKU_thesis_2003.pdf.
3. James David Schaffer, Larry J. Eshelman, and Daniel Offutt. Spurious correlations and premature convergence in genetic algorithms. In Bruce M. Spatz and Gregory J. E. Rawlins, editors, *Proceedings of the First Workshop on Foundations of Genetic Algorithms (FOGA'90), July 15–16, 1990, Bloomington, IN, USA*, pages 102–112, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.
4. J. Shekel. Test functions for multimodal search techniques. In *Fifth Annual Princeton Conference on Information Science and Systems*, pages 354–359, Princeton, NJ, USA, March 1971. Princeton University Press.
5. Antanas Žilinskas. Algorithm as 133: Optimization of one-dimensional multimodal functions. *Journal of the Royal Statistical Society: Series C – Applied Statistics*, 27(3):367–375, 1978. doi: 10.2307/2347182.
6. Kenneth Alan De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, Ann Arbor, MI, USA, August 1975. URL http://cs.gmu.edu/~eclab/kdj_thesis.html.
7. Jeffrey Horn and David Edward Goldberg. Genetic algorithm difficulty and the modality of the fitness landscape. In L. Darrell Whitley and Michael D. Vose, editors, *Proceedings of the Third Workshop on Foundations of Genetic Algorithms (FOGA 3), July 31–August 2, 1994, Estes Park, CO, USA*, pages 243–269, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-356-5.
8. Soraya Rana. *Examining the Role of Local Optima and Schema Processing in Genetic Search*. PhD thesis, Colorado State University, Department of Computer Science, GENITOR Research Group in Genetic Algorithms and Evolutionary Computation, Fort Collins, CO, USA, July 1999.
9. Larry J. Eshelman, Richard A. Caruana, and James David Schaffer. Biases in the crossover landscape. In James David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms (ICGA'89), June 4–7, 1989, Fairfax, VA, USA*, pages 10–19, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.
10. John Henry Holland. Genetic algorithms – computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand. *Scientific American*, 267(1):44–50, July 1992. URL http://www2.econ.iastate.edu/tesfatsi/holland.gaintro.htm.

Bibliography II

11. Ágoston E. Eiben and C. A. Schippers. On evolutionary exploration and exploitation. *Fundamenta Informaticae – Annales Societatis Mathematicae Polonae, Series IV*, 35(1-2):35–50, July–August 1998. doi: 10.3233/FI-1998-35123403. URL http://www.cs.vu.nl/~gusz/papers/FunInf98-Eiben-Schippers.ps.
12. Nitin Muttil and Shie-Yui Liong. Superior exploration–exploitation balance in shuffled complex evolution. *Journal of Hydraulic Engineering*, 130(12):1202–1205, December 2004. doi: 10.1061/(ASCE)0733-9429(2004)130:12(1202).
13. Heni Ben Amor and Achim Rettinger. Intelligent exploration for genetic algorithms: Using self-organizing maps in evolutionary computation. In Hans-Georg Beyer, Una-May O'Reilly, Dirk V. Arnold, Wolfgang Banzhaf, Christian Blum, Eric W. Bonabeau, Erick Cantú-Paz, Dipankar Dasgupta, Kalyanmoy Deb, James A. Foster, Edwin D. de Jong, Hod Lipson, Xavier Llorà, Spiros Mancoridis, Martin Pelikan, Günther R. Raidl, Terence Soule, Jean-Paul Watson, and Eckart Zitzler, editors, *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'05), June 25–27, 2005, Washington, DC, USA*, pages 1531–1538, New York, NY, USA, 2005. ACM Press. doi: 10.1145/1068009.1068250.
14. Kalyanmoy Deb. Genetic algorithms for optimization. KanGAL Report 2001002, Kanpur Genetic Algorithms Laboratory (KanGAL), Department of Mechanical Engineering, Indian Institute of Technology Kanpur (IIT), Kanpur, Uttar Pradesh, India, 2001. URL http://www.iitk.ac.in/kangal/papers/isna.ps.gz.
15. 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.
16. Fred W. Glover, Éric D. Taillard, and Dominique de Werra. A user's guide to tabu search. *Annals of Operations Research*, 41(1):3–28, March 1993. doi: 10.1007/BF02078647. Special issue on Tabu search.
17. Paola Festa and Mauricio G.C. Resende. An annotated bibliography of grasp. AT&T Labs Research Technical Report TD-5WYSEW, Florham Park, NJ, USA: AT&T Labs, February 29, 2004. URL http://www.research.att.com/~mgcr/grasp/gannbib/gannbib.html.
18. Thomas A. Feo and Mauricio G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, March 1995. doi: 10.1007/BF01096763. URL http://www.research.att.com/~mgcr/doc/gtut.ps.Z.
19. Tianshi Chen, Jun He, Guoliang Chen, and Xin Yao. Choosing selection pressure for wide-gap problems. *Theoretical Computer Science*, 411(6):926–934, February 6, 2010. doi: 10.1016/j.tcs.2009.12.014. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.158.8889.

20. Jun He and Xin Yao. From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms. *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, 6(5):495–511, October 2002. doi: 10.1109/TEVC.2002.800886. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.148.2275.
21. Tianshi Chen, Ke Tang, Guoliang Chen, and Xin Yao. A large population size can be unhelpful in evolutionary algorithms. *Theoretical Computer Science*, 436:54–70, June 8, 2012. doi: 10.1016/j.tcs.2011.02.016.
22. Erik van Nimwegen and James P. Crutchfield. Optimizing epochal evolutionary search: Population-size dependent theory. *Machine Learning*, 45(1):77–114, October 2001. doi: 10.1023/A:1012497308906.
23. Scott Kirkpatrick, C. 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://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.4175.
24. 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.
25. Dean Jacobs, Jan Prins, Peter Siegel, and Kenneth Wilson. Monte carlo techniques in code optimization. *ACM SIGMICRO Newsletter*, 13(4):143–148, December 1982. Proceedings of the 15th Annual Workshop on Microprogramming (MICRO 15), October 5–7, 1982, Palo Alto, CA, USA, New York, NY, USA: ACM.
26. Martin Pincus. Letter to the editor – a monte carlo method for the approximate solution of certain types of constrained optimization problems. *Operations Research*, 18(6):1225–1228, November–December 1970. doi: 10.1287/opre.18.6.1225.
27. John Henry Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI, USA, 1975. ISBN 0-472-08460-7.
28. Kalyanmoy Deb. Genetic algorithms for multimodal function optimization. Master's thesis, Tuscaloosa: Clearinghouse for Genetic Algorithms, University of Alabama, 1989. TCGA Report No. 89002.
29. David Edward Goldberg and Jon T. Richardson. Genetic algorithms with sharing for multimodal function optimization. In John J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms and their Applications (ICGA'87)*, pages 41–49, Cambridge, MA, USA: Massachusetts Institute of Technology (MIT), July 28–31, 1987. Mahwah, NJ, USA: Lawrence Erlbaum Associates, Inc. (LEA).

Bibliography IV

30. Kalyanmoy Deb and David Edward Goldberg. An investigation of niche and species formation in genetic function optimization. In James David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms (ICGA'89)*, pages 42–50, Fairfax, VA, USA: George Mason University (GMU), June 4–7, 1989. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
31. Bruno Sareni and Laurent Krähenbühl. Fitness sharing and niching methods revisited. *IEEE Transactions on Evolutionary Computation (TEVC)*, 2(3):97–106, 1998. doi: 10.1109/4235.735432.
32. Brad L. Miller and Michael J. Shaw. Genetic algorithms with dynamic niche sharing for multimodal function optimization. IlliGAL Report 95010, Urbana-Champaign, IL, USA: University of Illinois at Urbana-Champaign, Department of Computer Science, Department of General Engineering, Illinois Genetic Algorithms Laboratory (IlliGAL), December 1, 1995. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.3568`.
33. David Edward Goldberg, Kalyanmoy Deb, and Jeffrey Horn. Massive multimodality, deception, and genetic algorithms. In Reinhard Männer and Bernard Manderick, editors, *Proceedings of Parallel Problem Solving from Nature 2 (PPSN II)*, pages 37–48, Brussels, Belgium, September 28–30, 1992. Amsterdam, The Netherlands: Elsevier Science Publishers B.V. and Amsterdam, The Netherlands: North-Holland Scientific Publishers Ltd.
34. Thomas Weise. *Global Optimization Algorithms – Theory and Application*. it-weise.de (self-published), Germany, 2009. URL `http://www.it-weise.de/projects/book.pdf`.
35. Thomas Weise, Alexander Podlich, and Christian Gorldt. Solving real-world vehicle routing problems with evolutionary algorithms. In Raymond Chiong and Sandeep Dhakal, editors, *Natural Intelligence for Scheduling, Planning and Packing Problems*, volume 250 of *Studies in Computational Intelligence (SCI)*, chapter 2, pages 29–53. Springer-Verlag, Berlin/Heidelberg, October 2009. doi: 10.1007/978-3-642-04039-9_2.
36. Thomas Weise, Alexander Podlich, Kai Reinhard, Christian Gorldt, and Kurt Geihs. Evolutionary freight transportation planning. In Mario Giacobini, Penousal Machado, Anthony Brabazon, Jon McCormack, Stefano Cagnoni, Michael O'Neill, Gianni A. Di Caro, Ferrante Neri, Anikó Ekárt, Mike Preuß, Anna Isabel Esparcia-Alcázar, Franz Rothlauf, Muddassar Farooq, Ernesto Tarantino, Andreas Fink, and Shengxiang Yang, editors, *Applications of Evolutionary Computing – Proceedings of EvoWorkshops 2009: EvoCOMNET, EvoENVIRONMENT, EvoFIN, EvoGAMES, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, EvoNUM, EvoSTOC, EvoTRANSLOG, April 15–17, 2009, Tübingen, Germany*, volume 5484/2009 of *Lecture Notes in Computer Science (LNCS)*, pages 768–777, Berlin, Germany, 2009. Springer-Verlag GmbH. doi: 10.1007/978-3-642-01129-0_87.

37. Alan Pétrowski. A clearing procedure as a niching method for genetic algorithms. In Keisoku Jidō and Seigyo Gakkai, editors, *Proceedings of IEEE International Conference on Evolutionary Computation (CEC'96), May 20–22, 1996, Nagoya, Japan*, pages 798–803, Los Alamitos, CA, USA, 1996. IEEE Computer Society Press. ISBN 0-7803-2902-3. doi: 10.1109/ICEC.1996.542703.
38. Alan Pétrowski. An efficient hierarchical clustering technique for speciation. Technical report, Institut National des Télécommunications, Evry Cedex, France, 1997.
39. Simon Ronald, John Asenstorfer, and Millist Vincent. Representational redundancy in evolutionary algorithms. In *Second IEEE International Conference on Evolutionary Computation (CEC'95)*, volume 2, pages 631–637, Perth, WA, Australia: University of Western Australia, November 29–December 1, 1995. Los Alamitos, CA, USA: IEEE Computer Society Press. doi: 10.1109/ICEC.1995.487457.
40. Paul J. Darwen and Xin Yao. Every niching method has its niche: Fitness sharing and implicit sharing compared. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Proceedings The 4th International Conference on Parallel Problem Solving from Nature PPSN IV, International Conference on Evolutionary Computation, September 22–26, 1996, Berlin, Germany*, volume 1141 of *Lecture Notes in Computer Science (LNCS)*, pages 398–407. Springer, 1996. ISBN 3-540-61723-X. doi: 10.1007/3-540-61723-X_1004.
41. Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, Technische Universität Berlin, Berlin, Germany, 1971–1973.
42. Ingo Rechenberg. *Evolutionsstrategie '94*, volume 1 of *Werkstatt Bionik und Evolutionstechnik*. Frommann-Holzboog Verlag, Bad Cannstadt, Stuttgart, Baden-Württemberg, Germany, 1994. ISBN 3-7728-1642-8.
43. Justinian P. Rosca and Dana H. Ballard. Causality in genetic programming. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA'95)*, pages 256–263, 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=10.1.1.26.2503.

44. Jean-Baptiste Pierre Antoine de Monet, Chevalier de Lamarck. *Philosophie zoologique – ou Exposition des considérations relatives à l'histoire naturelle des Animaux; à la diversité de leur organisation et des facultés qu'ils en obtiennent; ....* Paris, France: Dentu and Paris, France: J. B. Baillière Liberaire, 1809. ISBN 1412116465 and 9781412116466. URL http://www.lamarck.cnrs.fr/ice/ice_book_detail.php?type=text&bdd=lamarck&table=ouvrages_lamarck&bookId=29. Philosophie zoologique – ou Exposition des considérations relatives à l'histoire naturelle des Animaux; à la diversité de leur organisation et des facultés qu'ils en obtiennent; aux causes physiques qui maintiennent en eux la vie et donnent lieu aux mouvements qu'ils exécutant; enfin, à celles qui produisent les unes le sentiment, et les autres l'intelligence de ceux qui en sont doués.
45. L. Darrell Whitley, V. Scott Gordon, and Keith E. Mathias. Lamarckian evolution, the Baldwin effect and function optimization. In Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer, editors, *Proceedings of the Third Conference on Parallel Problem Solving from Nature; International Conference on Evolutionary Computation (PPSN III), October 9–14, 1994, Jerusalem, Israel*, volume 866/1994 of *Lecture Notes in Computer Science (LNCS)*, pages 5–15, Berlin, Germany, 1994. Springer-Verlag GmbH. ISBN 0387584846. doi: 10.1007/3-540-58484-6_245. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.2428.
46. James Mark Baldwin. A new factor in evolution. *The American Naturalist*, 30(354):441–451, June 1896. URL http://www.brocku.ca/MeadProject/Baldwin/Baldwin_1896_h.html.
47. Geoffrey Everest Hinton and Steven J. Nowlan. How learning can guide evolution. *Complex Systems*, 1(3), 1987. URL https://www.complex-systems.com/abstracts/v01_i03_a06/.
48. Geoffrey E. Hinton and Steven J. Nowlan. How learning can guide evolution. In Richard K. Belew and Melanie Mitchell, editors, *Adaptive Individuals in Evolving Populations: Models and Algorithms*, volume 26 of *Santa Fe Institue Studies in the Sciences of Complexity*, pages 447–454. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. and Boulder, CO, USA: Westview Press, January 15, 1996.
49. Pablo Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech Concurrent Computation Program C3P 826, California Institute of Technology (Caltech), Caltech Concurrent Computation Program (C3P), Pasadena, CA, USA, 1989. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.27.9474.

50. Michael G. Norman and Pablo Moscato. A competitive and cooperative approach to complex combinatorial search. In *Proceedings of the 20th Informatics and Operations Research Meeting (20th Jornadas Argentinas e Informática e Investigación Operativa) (JAIIO'91)*, pages 3.15–3.29, Buenos Aires, Argentina: Centro Cultural General San Martín, August 20–23, 1991. Also published as Technical Report Caltech Concurrent Computation Program, Report. 790, California Institute of Technology, Pasadena, California, USA, 1989.
51. Diana Holstein and Pablo Moscato. Memetic algorithms using guided local search: A case study. In 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, pages 235–244. Maidenhead, England, UK: McGraw-Hill Ltd., May 1999.
52. Luciana Buriol, Paulo M. França, and Pablo Moscato. A new memetic algorithm for the asymmetric traveling salesman problem. *Journal of Heuristics*, 10(5):483–506, September 2004. doi: 10.1023/B:HEUR.0000045321.59202.52.
53. Maria J. Blesa, Pablo Moscato, and Fatos Xhafa. A memetic algorithm for the minimum weighted k-cardinality tree subgraph problem. In Mauricio G.C. Resende, Jorge Pinho de Sousa, and Ana Viana, editors, *4th Metaheuristics International Conference – Metaheuristics: Computer Decision-Making (MIC'01)*, volume 86 of *Applied Optimization*, pages 85–90, Porto, Portugal, July 16–20, 2001. Berlin, Germany: Springer-Verlag GmbH. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.5376`.
54. 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`.
55. 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`.

Bibliography VIII

56. Jason Digalakis and Konstantinos Margaritis. A parallel memetic algorithm for solving optimization problems. In Mauricio G.C. Resende, Jorge Pinho de Sousa, and Ana Viana, editors, *4th Metaheuristics International Conference – Metaheuristics: Computer Decision-Making (MIC'01)*, volume 86 of *Applied Optimization*, pages 121–125, Porto, Portugal, July 16–20, 2001. Berlin, Germany: Springer-Verlag GmbH. URL `http://citeseer.ist.psu.edu/digalakis01parallel.html`.
57. 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`.
58. Natalio Krasnogor and James E. Smith. A tutorial for competent memetic algorithms: Model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, 9(5):474–488, October 2005. doi: 10.1109/TEVC.2005.850260. URL `http://www.cs.nott.ac.uk/~nxk/PAPERS/IEEE-TEC-lastVersion.pdf`.
59. David H. Ackley. *A Connectionist Machine for Genetic Hillclimbing*. PhD thesis, Pittsburgh, PA, USA: Carnegy Mellon University (CMU), August 31, 1987. URL `http://books.google.de/books?id=3ttfAAAAMAAJ`.
60. David Edward Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989. ISBN 0-201-15767-5.
61. Martina Gorges-Schleuter. Asparagos: An asynchronous parallel genetic optimization strategy. In James David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms (ICGA'89)*, pages 422–427, Fairfax, VA, USA: George Mason University (GMU), June 4–7, 1989. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
62. Heinz Mühlenbein. Parallel genetic algorithms, population genetics and combinatorial optimization. In Jörg D. Becker, Ignaz Eisele, and F. W. Mündemann, editors, *Parallelism, Learning, Evolution: Workshop on Evolutionary Models and Strategies (Neubiberg, Germany, 1989-03-10/11) and Workshop on Parallel Processing: Logic, Organization, and Technology (Wildbad Kreuth, Germany, 1989-07-24 to 28) (WOPPLOT'89)*, volume 565/1991 of *Lecture Notes in Computer Science (LNCS)*, pages 398–406, Germany, 1989. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/3-540-55027-5_23.
63. Heinz Mühlenbein. Parallel genetic algorithms, population genetics and combinatorial optimization. In James David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms (ICGA'89)*, pages 416–421, Fairfax, VA, USA: George Mason University (GMU), June 4–7, 1989. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

64. Heinz Mühlenbein. How genetic algorithms really work – i. mutation and hillclimbing. In Reinhard Männer and Bernard Manderick, editors, *Proceedings of Parallel Problem Solving from Nature 2 (PPSN II)*, pages 15–26, Brussels, Belgium, September 28–30, 1992. Amsterdam, The Netherlands: Elsevier Science Publishers B.V. and Amsterdam, The Netherlands: North-Holland Scientific Publishers Ltd. URL http://muehlenbein.org/mut92.pdf.
65. Donald E. Brown, Christopher L. Huntley, and Andrew R. Spillane. A parallel genetic heuristic for the quadratic assignment problem. In James David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms (ICGA'89)*, pages 406–415, Fairfax, VA, USA: George Mason University (GMU), June 4–7, 1989. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
66. Lawrence Davis, editor. *Handbook of Genetic Algorithms*. VNR Computer Library. Stamford, CT, USA: Thomson Publishing Group, Inc. and New York, NY, USA: Van Nostrand Reinhold Co., January 1991. ISBN 0-442-00173-8, 1850328250, 978-0-442-00173-5, and 978-1850328254. URL http://books.google.de/books?id=vTG5PAAACAAJ.
67. Ko-Hsin Liang, Xin Yao, and Charles S. Newton. Evolutionary search of approximated n-dimensional landscapes. *International Journal of Knowledge-Based and Intelligent Engineering Systems*, 4(3):172–183, June 2000. URL http://sclab.yonsei.ac.kr/courses/03EC/liang00evolutionary.pdf.
68. Albert Donally Bethke. *Genetic Algorithms as Function Optimizers*. PhD thesis, Ann Arbor, MI, USA: University of Michigan, 1980. National Science Foundation Grant No. MCS76-04297.
69. Gunar E. Liepins and Michael D. Vose. Deceptiveness and genetic algorithm dynamics. In Bruce M. Spatz and Gregory J. E. Rawlins, editors, *Proceedings of the First Workshop on Foundations of Genetic Algorithms (FOGA'90)*, pages 36–50, Bloomington, IN, USA: Indiana University, Bloomington Campus, July 15–18, 1990. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. URL http://www.osti.gov/bridge/servlets/purl/6445602-CfqU6M/.
70. Thorsten Schnier and Xin Yao. Using multiple representations in evolutionary algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'00)*, volume 1, pages 479–486, La Jolla, CA, USA: La Jolla Marriott Hotel, July 16–19, 2000. Piscataway, NJ, USA: IEEE Computer Society. doi: 10.1109/CEC.2000.870335. URL http://www.cs.bham.ac.uk/~txs/publications/2000/SchnierYaoCEC2000.pdf.
71. Shane Legg, Marcus Hutter, and Akshat Kumar. Tournament versus fitness uniform selection. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'04)*, pages 2144–2151, Portland, OR, USA, June 20–23, 2004. Los Alamitos, CA, USA: IEEE Computer Society Press. doi: 10.1109/CEC.2004.1331162. URL http://arxiv.org/abs/cs/0403038v1.

72. Marcus Hutter. Fitness uniform selection to preserve genetic diversity. In David B. Fogel, Mohamed A. El-Sharkawi, Xin Yao, Hitoshi Iba, Paul Marrow, and Mark Shackleton, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'02), 2002 IEEE World Congress on Computation Intelligence (WCCI'02)*, volume 1-2, pages 783–788, Honolulu, HI, USA: Hilton Hawaiian Village Hotel (Beach Resort & Spa), May 12–17, 2002. Piscataway, NJ, USA: IEEE Computer Society, Los Alamitos, CA, USA: IEEE Computer Society Press. doi: 10.1109/CEC.2002.1007025. URL http://arxiv.org/abs/cs/0103015. Also: Technical Report IDSIA-01-01, 17 January 2001.
73. Marcus Hutter and Shane Legg. Fitness uniform optimization. *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, 10(5):568–589, October 2006. doi: 10.1109/TEVC.2005.863127. URL http://arxiv.org/abs/cs/0610126v1.
74. Shane Legg and Marcus Hutter. Fitness uniform deletion: A simple way to preserve diversity. In Hans-Georg Beyer, Una-May O'Reilly, Dirk V. Arnold, Wolfgang Banzhaf, Christian Blum, Eric W. Bonabeau, Erick Cantú-Paz, Dipankar Dasgupta, Kalyanmoy Deb, James A. Foster, Edwin D. de Jong, Hod Lipson, Xavier Llorà, Spiros Mancoridis, Martin Pelikan, Günther R. Raidl, Terence Soule, Jean-Paul Watson, and Eckart Zitzler, editors, *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'05)*, pages 1271–1278, Washington, DC, USA: Loews L'Enfant Plaza Hotel, June 25–27, 2005. New York, NY, USA: ACM Press. doi: 10.1145/1068009.1068216. URL http://arxiv.org/abs/cs/0504035v1.
75. Joel Lehman and Kenneth Owen Stanley. Exploiting open-endedness to solve problems through the search for novelty. In Seth Bullock, Jason Noble, Richard A. Watson, and Mark A. Bedau, editors, *Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems (Artificial Life XI)*, pages 329–336, Winchester, Hampshire, UK, August 5–8, 2008. Cambridge, MA, USA: MIT Press. URL http://eplex.cs.ucf.edu/papers/lehman_alife08.pdf.
76. Joel Lehman and Kenneth Owen Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223, Summer 2011. doi: 10.1162/EVCO_a_00025. URL http://eplex.cs.ucf.edu/papers/lehman_ecj10.pdf.
77. Joel Lehman and Kenneth Owen Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'11)*, pages 211–218, Dublin, Ireland, July 12–16, 2011. doi: 10.1145/2001576.2001606. URL http://eplex.cs.ucf.edu/papers/lehman_gecco11.pdf.

78. Thomas Weise, Mingxu Wan, Ke Tang, Pu Wang, Alexandre Devert, and Xin Yao. Frequency fitness assignment. *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, 18(2):226–243, April 2014. doi: 10.1109/TEVC.2013.2251885.
79. Yuval Davidor. Epistasis variance: A viewpoint on ga-hardness. In Bruce M. Spatz and Gregory J. E. Rawlins, editors, *Proceedings of the First Workshop on Foundations of Genetic Algorithms (FOGA'90)*, pages 23–35, Bloomington, IN, USA: Indiana University, Bloomington Campus, July 15–18, 1990. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
80. Lee Altenberg. Nk fitness landscapes. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, Computational Intelligence Library, chapter B2.7.2. 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. URL `http://www.cmi.univ-mrs.fr/~pardoux/LeeNKFL.pdf`.
81. Bart Naudts and Alain Verschoren. Epistasis on finite and infinite spaces. In *8th International Conference on Systems Research, Informatics and Cybernetics (InterSymp'96)*, pages 19–23, Baden-Baden, Baden-Württemberg, Germany, August 14–18, 1996. Tecumseh, ON, Canada: International Institute for Advanced Studies in Systems Research and Cybernetic (IIAS). URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.6455`.
82. Colin R. Reeves and Christine C. Wright. Epistasis in genetic algorithms: An experimental design perspective. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA'95)*, pages 217–224, 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=10.1.1.39.29`.
83. William Bateson. *Mendel's Principles of Heredity*. Kessinger Publishing's® Rare Reprints. Cambridge, UK: Cambridge University Press, 1909. ISBN 1428648194 and 9781428648197. URL `http://books.google.de/books?id=WWZfDQljn8gC`.
84. Sir Ronald Aylmer Fisher. The correlations between relatives on the supposition of mendelian inheritance. *Philosophical Transactions of the Royal Society of Edinburgh*, 52:399–433, October 1, 1918. URL `http://www.library.adelaide.edu.au/digitised/fisher/9.pdf`.
85. Patrick C. Phillips. The language of gene interaction. *Genetics*, 149:1167–1171, July 1998. URL `http://www.genetics.org/content/149/3/1167`.
86. Jay L. Lush. Progeny test and individual performance as indicators of an animal's breeding value. *Journal of Dairy Science (JDS)*, 18(1):1–19, January 1935. URL `http://jds.fass.org/cgi/reprint/18/1/1`.

87. Ke Tang, Xin Yao, Ponnuthurai Nagaratnam Suganthan, Cara MacNish, Ying-Ping Chen, Chih-Ming Chen, and Zhenyu Yang. Benchmark functions for the cec'2008 special session and competition on large scale global optimization. Technical report, Hefei, Anhui, China: University of Science and Technology of China (USTC), School of Computer Science and Technology, Nature Inspired Computation and Applications Laboratory (NICAL), 2007. URL http://sci2s.ugr.es/programacion/workshop/Tech.Report.CEC2008.LSGO.pdf.
88. George Hadley. *Nonlinear and Dynamics Programming*. World Student. Addison-Wesley Professional, Reading, MA, USA, 1964. ISBN 0201026643.
89. Domingo Ortiz-Boyer, César Hervás-Martínez, and Carlos A. Reyes García. Cixl2: A crossover operator for evolutionary algorithms based on population features. *Journal of Artificial Intelligence Research (JAIR)*, 24:1–48, July 2005. URL http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume24/ortizboyer05a-html/Ortiz-Boyer.html.
90. Ke Tang, Xiaodong Li, Ponnuthurai Nagaratnam Suganthan, Zhenyu Yang, and Thomas Weise. Benchmark functions for the cec'2010 special session and competition on large-scale global optimization. Technical report, University of Science and Technology of China (USTC), School of Computer Science and Technology, Nature Inspired Computation and Applications Laboratory (NICAL), Hefei, Anhui, China, January 8 2010.
91. Thomas Weise and Ke Tang. Evolving distributed algorithms with genetic programming. *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, 16(2):242–265, April 2012. doi: 10.1109/TEVC.2011.2112666.
92. Thomas Weise, Michael Zapf, and Kurt Geihs. Rule-based genetic programming. In *Proceedings of the 2nd International Conference on Bio-Inspired Models of Network, Information, and Computing Systems (BIONETICS'07)*, pages 8–15, Budapest, Hungary: Radisson SAS Beke Hotel, December 10–13, 2007. Piscataway, NJ, USA: IEEE Computer Society. doi: 10.1109/BIMNICS.2007.4610073.
93. Thomas Weise and Michael Zapf. Evolving distributed algorithms with genetic programming: Election. In Lihong Xu, Erik D. Goodman, and Yongsheng Ding, editors, *Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC'09)*, pages 577–584, Shanghai, China: Hua-Ting Hotel & Towers, June 12–14, 2009. New York, NY, USA: ACM Press. doi: 10.1145/1543834.1543913.
94. Thomas Weise. *Evolving Distributed Algorithms with Genetic Programming*. PhD thesis, Kassel, Hesse, Germany: University of Kassel, Fachbereich 16: Elektrotechnik/Informatik, Distributed Systems Group, May 4, 2009. Won the Dissertation Award of The Association of German Engineers (Verein Deutscher Ingenieure, VDI).

Bibliography XIII

95. Masaya Shinkai, Arturo Hernández Aguirre, and Kiyoshi Tanaka. Mutation strategy improves gas performance on epistatic problems. In David B. Fogel, Mohamed A. El-Sharkawi, Xin Yao, Hitoshi Iba, Paul Marrow, and Mark Shackleton, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'02), 2002 IEEE World Congress on Computation Intelligence (WCCI'02)*, volume 1, pages 968–973, Honolulu, HI, USA: Hilton Hawaiian Village Hotel (Beach Resort & Spa), May 12–17, 2002. Piscataway, NJ, USA: IEEE Computer Society, Los Alamitos, CA, USA: IEEE Computer Society Press. doi: 10.1109/CEC.2002.1007056.
96. Georges Raif Harik. *Learning Gene Linkage to Efficiently Solve Problems of Bounded Difficulty using Genetic Algorithms*. PhD thesis, University of Michigan, Ann Arbor, MI, USA, 1997. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.54.7092`.
97. Masaharu Munetomo and David Edward Goldberg. Linkage identification by non-monotonicity detection for overlapping functions. *Evolutionary Computation*, 7(4):377–398, Winter 1999. doi: 10.1162/evco.1999.7.4.377.
98. Ying-Ping Chen. *Extending the Scalability of Linkage Learning Genetic Algorithms – Theory & Practice*, volume 190/2006 of *Studies in Fuzziness and Soft Computing book series (STUDFUZZ)*. Springer-Verlag GmbH, Berlin, Germany, 2004. ISBN 3-540-28459-1. doi: 10.1007/b102053. PhD thesis at Urbana-Champaign, IL, USA: University of Illinois under the advise of David Edward Goldberg.
99. Kalyanmoy Deb, Ankur Sinha, and Saku Kukkonen. Multi-objective test problems, linkages, and evolutionary methodologies. In Maarten Keijzer and Mike Cattolico, editors, *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'06)*, pages 1141–1148, Seattle, WA, USA: Renaissance Seattle Hotel, July 8–12, 2006. New York, NY, USA: ACM Press. doi: 10.1145/1143997.1144179.
100. David Edward Goldberg, Kalyanmoy Deb, and Bradley Korb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5):493–530, 1989. URL `http://www.complex-systems.com/pdf/03-5-5.pdf`.
101. Martin Pelikan, David Edward Goldberg, and Erick Cantú-Paz. BOA: The bayesian optimization algorithm. In Wolfgang Banzhaf, Jason M. Daida, Ágoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark J. Jakiela, and Robert Elliott Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99), July 13–17, 1999, Orlando, FL, USA*, pages 525–532, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-611-4.
102. Erick Cantú-Paz, Martin Pelikan, and David Edward Goldberg. Linkage problem, distribution estimation, and bayesian networks. *Evolutionary Computation*, 8(3):311–340, Autumn 2000. doi: 10.1162/106365600750078808. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.236`.

103. Wenxiang Chen, Thomas Weise, Zhenyu Yang, and Ke Tang. Large-scale global optimization using cooperative coevolution with variable interaction learning. In Robert Schaefer, Carlos Cotta, Joanna Kołodziej, and Günter Rudolph, editors, *Proceedings of the 11th International Conference on Parallel Problem Solving From Nature, (PPSN'10), Part 2, September 11–15, 2010, Kraków, Poland*, volume 6239 of *Lecture Notes in Computer Science (LNCS)*, pages 300–309, Berlin, Germany, 2010. Springer-Verlag GmbH. doi: 10.1007/978-3-642-15871-1_31.
104. Mitchell A. Potter and Kenneth Alan De Jong. A cooperative coevolutionary approach to function optimization. In Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer, editors, *Proceedings of the Third Conference on Parallel Problem Solving from Nature; International Conference on Evolutionary Computation (PPSN III)*, volume 866/1994 of *Lecture Notes in Computer Science (LNCS)*, pages 249–257, Jerusalem, Israel, October 9–14, 1994. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/3-540-58484-6_269. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.54.7033.
105. Mitchell A. Potter and Kenneth Alan De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, Spring 2000. doi: 10.1162/106365600568086. URL http://mitpress.mit.edu/journals/pdf/evco_8_1_1_0.pdf.
106. Richard Ernest Bellman. *Dynamic Programming*. Dover Books on Mathematics. Princeton University Press, Princeton, NJ, USA, 1957. ISBN 0486428095.
107. Richard Ernest Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton, NJ, USA: Princeton University Press, 1961. ISBN 0691079013 and 978-0691079011. URL http://books.google.de/books?id=8wY3PAAACAAJ.
108. Conor Ryan, John James Collins, and Michael O'Neill. Grammatical evolution: Evolving programs for an arbitrary language. In Wolfgang Banzhaf, Riccardo Poli, Marc Schoenauer, and Terence Claus Fogarty, editors, *Proceedings of the First European Workshop on Genetic Programming (EuroGP'98), April 14–15, 1998, Paris, France*, volume 1391/1998 of *Lecture Notes in Computer Science (LNCS)*, pages 83–95, Berlin, Germany, 1998. Springer-Verlag GmbH. doi: 10.1007/BFb0055930. URL http://www.grammatical-evolution.org/papers/eurogp98.ps.
109. Michael O'Neill and Conor Ryan. *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*, volume 4 of *Genetic Programming Series*. New York, NY, USA: Springer Science+Business Media, Inc., 2003. ISBN 1402074441 and 978-1-4020-7444-8. URL http://books.google.de/books?id=yVx5dg_opRIC.
110. Alexandre Devert, Thomas Weise, and Ke Tang. A study on scalable representations for evolutionary optimization of ground structures. *Evolutionary Computation*, 20(3):453–472, Fall 2012. doi: 10.1162/EVCO_a_00054. URL http://www.marmakoide.org/download/publications/devweita-ecj-preprint.pdf.

Bibliography XV

111. Alexandre Devert. *Building Processes Optimization: Toward an Artificial Ontogeny based Approach*. PhD thesis, Université Paris-Sud, Ecole Doctorale d'Informatique and Orsay, France: Institut National de Recherche en Informatique et en Automatique (INRIA), Centre de Recherche Saclay – Île-de-France, Paris, France, May 2009. Advisors: Marc Schoenauer and Nicolas Bredèche.
112. Phil Husbands and Frank Mill. Simulated co-evolution as the mechanism for emergent planning and scheduling. In Richard K. Belew and Lashon Bernard Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA'91)*, pages 264–270, San Diego, CA, USA: University of California (UCSD), July 13–16, 1991. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. URL http://www.informatics.sussex.ac.uk/users/philh/pubs/icga91Husbands.pdf.
113. Zhenyu Yang, Ke Tang, and Xin Yao. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences – Informatics and Computer Science Intelligent Systems Applications: An International Journal*, 178(15), August 1, 2008. doi: 10.1016/j.ins.2008.02.017. URL http://nical.ustc.edu.cn/papers/yangtangyao_ins.pdf.
114. Antonio LaTorre, José María Peña, Santiago Muelas, and Manuel Zaforas. Hybrid evolutionary algorithms for large scale continuous problems. In Franz Rothlauf, Günther R. Raidl, Anna Isabel Esparcia-Alcázar, Ying-Ping Chen, Gabriela Ochoa, Ender Ozcan, Marc Schoenauer, Anne Auger, Hans-Georg Beyer, Nikolaus Hansen, Steffen Finck, Raymond Ros, L. Darrell Whitley, Garnett Wilson, Simon Harding, William Benjamin Langdon, Man Leung Wong, Laurence D. Merkle, Frank W. Moore, Sevan G. Ficici, William Rand, Rick L. Riolo, Nawwaf Kharma, William R. Buckley, Julian Francis Miller, Kenneth Owen Stanley, Jaume Bacardit i Peñarroya, Will N. Browne, Jan Drugowitsch, Nicola Beume, Mike Preuß, Stephen Frederick Smith, Stefano Cagnoni, Alexandru Floares, Aaron Baughman, Steven Matt Gustafson, Maarten Keijzer, Arthur Kordon, and Clare Bates Congdon, editors, *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO'09)*, pages 1863–1864, Montréal, QC, Canada: Delta Centre-Ville Hotel, July 8–12, 2009. New York, NY, USA: Association for Computing Machinery (ACM). doi: 10.1145/1569901.1570205.
115. Fei Peng, Ke Tang, Guoliang Chen, and Xin Yao. Population-based algorithm portfolios for numerical optimization. *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, 14(5):782–800, March 29, 2010. doi: 10.1109/TEVC.2010.2040183.
116. Gene M. Amdahl. Validity of the single processor approach to achieving large-scale computing capabilities. In *American Federation of Information Processing Societies: Proceedings of the Spring Joint Computer Conference (AFIPS), April 18–20, 167, Atlantic City, NJ, USA*, pages 483–485, New York, NY, USA, 1967. doi: 10.1145/1465482.1465560. URL http://www-inst.eecs.berkeley.edu/~n252/paper/Amdahl.pdf.

117. Simon Harding and Wolfgang Banzhaf. Fast genetic programming on gpus. In Marc Ebner, Michael O'Neill, Anikó Ekárt, Leonardo Vanneschi, and Anna Isabel Esparcia-Alcázar, editors, *Proceedings of the 10th European Conference on Genetic Programming (EuroGP'07)*, volume 4445/2007 of *Lecture Notes in Computer Science (LNCS)*, pages 90–101, València, Spain, April 11–13, 2007. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/978-3-540-71605-1_9. URL http://www.cs.mun.ca/~banzhaf/papers/eurogp07.pdf.
118. William Benjamin Langdon. A simd interpreter for genetic programming on gpu graphics cards. Computer Science Technical Report CSM-470, Wivenhoe Park, Colchester, Essex, UK: University of Essex, Departments of Mathematical and Biological Sciences, July 3, 2007. URL http://cswww.essex.ac.uk/technical-reports/2007/csm_470.pdf.
119. William Benjamin Langdon and Wolfgang Banzhaf. A simd interpreter for genetic programming on gpu graphics cards. In Michael O'Neill, Leonardo Vanneschi, Steven Matt Gustafson, Anna Isabel Esparcia-Alcázar, Ivanoe de Falco, Antonio Della Cioppa, and Ernesto Tarantino, editors, *Genetic Programming – Proceedings of the 11th European Conference on Genetic Programming (EuroGP'08)*, volume 4971/2008 of *Lecture Notes in Computer Science (LNCS)*, pages 73–85, Naples, Italy, March 26–28, 2008. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/978-3-540-78671-9_7. URL http://www.cs.bham.ac.uk/~wbl/biblio/gp-html/langdon_2008_eurogp.html.
120. Weihang Zhu. A study of parallel evolution strategy – pattern search on a gpu computing platform. In Lihong Xu, Erik D. Goodman, and Yongsheng Ding, editors, *Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC'09)*, pages 765–771, Shanghai, China: Hua-Ting Hotel & Towers, June 12–14, 2009. New York, NY, USA: ACM Press. doi: 10.1145/1543834.1543939.
121. Shigeyoshi Tsutsui and Yoshiji Fujimoto. Solving quadratic assignment problems by genetic algorithms with gpu computation: A case study. In Franz Rothlauf, Günther R. Raidl, Anna Isabel Esparcia-Alcázar, Ying-Ping Chen, Gabriela Ochoa, Ender Ozcan, Marc Schoenauer, Anne Auger, Hans-Georg Beyer, Nikolaus Hansen, Steffen Finck, Raymond Ros, L. Darrell Whitley, Garnett Wilson, Simon Harding, William Benjamin Langdon, Man Leung Wong, Laurence D. Merkle, Frank W. Moore, Sevan G. Ficici, William Rand, Rick L. Riolo, Nawwaf Kharma, William R. Buckley, Julian Francis Miller, Kenneth Owen Stanley, Jaume Bacardit i Peñarroya, Will N. Browne, Jan Drugowitsch, Nicola Beume, Mike Preuß, Stephen Frederick Smith, Stefano Cagnoni, Alexandru Floares, Aaron Baughman, Steven Matt Gustafson, Maarten Keijzer, Arthur Kordon, and Clare Bates Congdon, editors, *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO'09)*, pages 2523–2530, Montréal, QC, Canada: Delta Centre-Ville Hotel, July 8–12, 2009. New York, NY, USA: Association for Computing Machinery (ACM). doi: 10.1145/1570256.1570355. URL http://www2.hannan-u.ac.jp/~tsutsui/ps/gecco/wk3006-tsutsui.pdf.

122. Kamil Rocki and Reiji Suda. An efficient gpu implementation of a multi-start tsp solver for large problem instances. In Terence Soule and Jason H. Moore, editors, *Companion Material Proceedings Genetic and Evolutionary Computation Conference (GECCO'12)*, pages 1441–1442, Philadelphia, PA, USA: Doubletree by Hilton Hotel Philadelphia Center City, July 7–11, 2012. New York, NY, USA: Association for Computing Machinery (ACM). doi: 10.1145/2330784.2330978.
123. Thomas Weise and Kurt Geihs. Dgpf – an adaptable framework for distributed multi-objective search algorithms applied to the genetic programming of sensor networks. 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 157–166, Ljubljana, Slovenia: Jožef Stefan International Postgraduate School, October 9–10, 2006. Ljubljana, Slovenia: Jožef Stefan Institute.
124. Thomas Weise, Kurt Geihs, and Philipp Andreas Baer. Genetic programming for proactive aggregation protocols. In Bartłomiej Beliczyński, Andrzej Dzieliński, Marcin Iwanowski, and Bernardete Ribeiro, editors, *Proceedings of the 8th International Conference on Adaptive and Natural Computing Algorithms, Part I (ICANNGA'07)*, volume 4431/2007 of *Lecture Notes in Computer Science (LNCS)*, pages 167–173, Warsaw, Poland: Warsaw University, April 11–17, 2007. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/978-3-540-71618-1_19.
125. 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.
126. Thomas Weise, Raymond Chiong, and Ke Tang. Evolutionary optimization: Pitfalls and booby traps. *Journal of Computer Science and Technology (JCST)*, 27:907–936, September 2012. doi: 10.1007/s11390-012-1274-4.
127. Thomas Weise, Michael Zapf, Raymond Chiong, and Antonio Jesús Nebro Urbaneja. Why is optimization difficult? In Raymond Chiong, editor, *Nature-Inspired Algorithms for Optimisation*, volume 193/2009 of *Studies in Computational Intelligence (SCI)*, chapter 1, pages 1–50. Springer-Verlag, Berlin/Heidelberg, April 2009. ISBN 978-3-642-00266-3. doi: 10.1007/978-3-642-00267-0_1.