Metaheuristic Optimization

1. Introduction

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

website

Transportation Planning: Task

- In this unit, we want to get a rough feeling about what *optimization* is.

Transportation Planning: Task

- In this unit, we want to get a rough feeling about what *optimization* is.
- So let us start by looking at some examples for optimization problems.

Section Outline

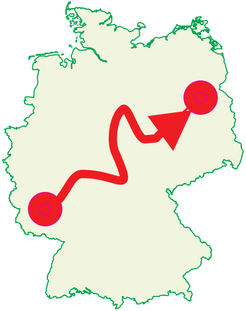
Transportation Planning: Task

- Build a system which tells a logistics company what it needs to do to fulfill all transportation orders at minimum costs. [1–5]

Transportation Planning: Task

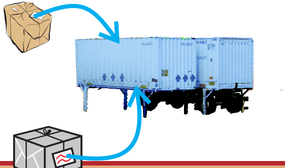
- Build a system which tells a logistics company what it needs to do to fulfill all transportation orders at minimum costs. [1–5]
- What does this mean?

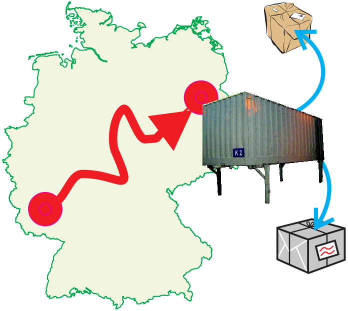
- Build a system which tells a logistics company what it needs to do to fulfill all transportation orders at minimum costs. [1–5]
  1. Find routes on the map



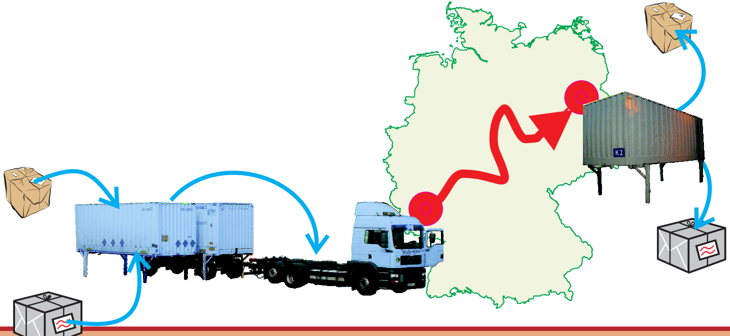
Transportation Planning: Task

- Build a system which tells a logistics company what it needs to do to fulfill all transportation orders at minimum costs. [1–5]
  1. Find routes on the map and assignments of orders to containers



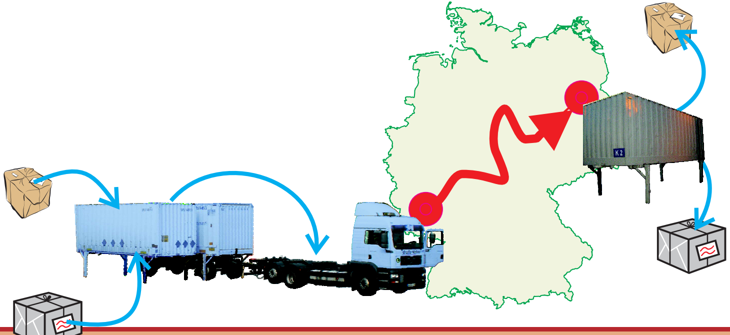


- Build a system which tells a logistics company what it needs to do to fulfill all transportation orders at minimum costs. [1–5]
  1. Find routes on the map and assignments of orders to containers and containers to trucks

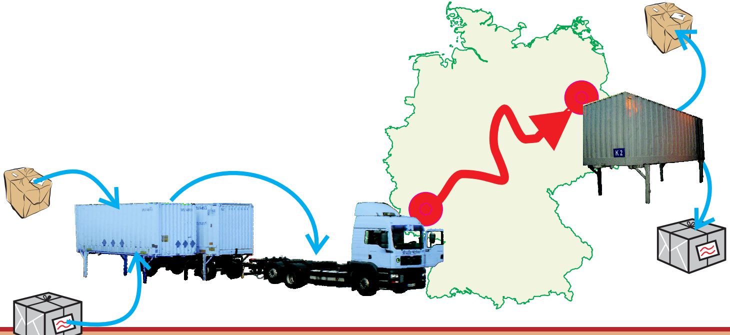


Transportation Planning: Task

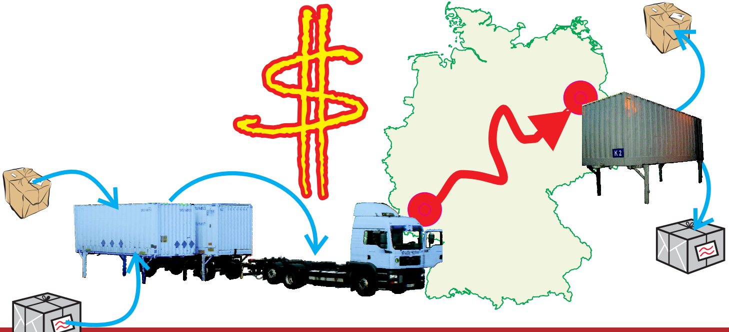
- Build a system which tells a logistics company what it needs to do to fulfill all transportation orders at minimum costs. [1–5]
  1. Find routes on the map and assignments of orders to containers and containers to trucks/trains



- Build a system which tells a logistics company what it needs to do to fulfill all transportation orders at minimum costs. [1–5]
  1. Find routes on the map and assignments of orders to containers and containers to trucks/trains which minimize the undelivered orders

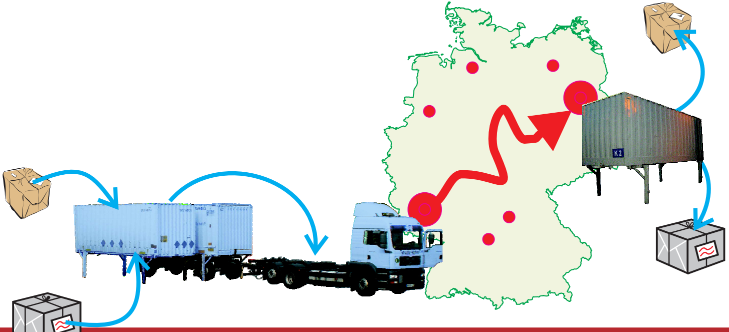


- Build a system which tells a logistics company what it needs to do to fulfill all transportation orders at minimum costs. [1–5]
  1. Find routes on the map and assignments of orders to containers and containers to trucks/trains which minimize the undelivered orders and the total distance for



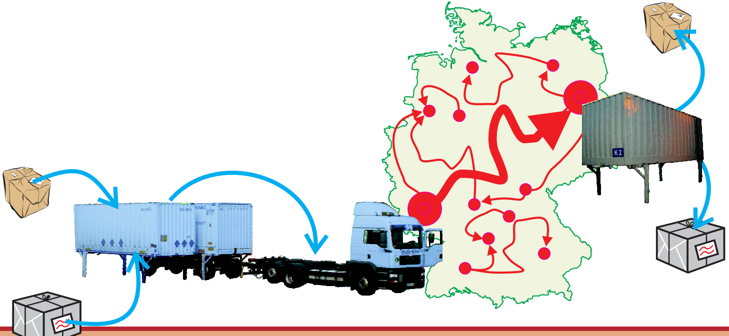
Transportation Planning: Task

- Build a system which tells a logistics company what it needs to do to fulfill all transportation orders at minimum costs. [1–5]
  1. Find routes on the map and assignments of orders to containers and containers to trucks/trains which minimize the undelivered orders and the total distance for...
  2. multiple depots



Transportation Planning: Task

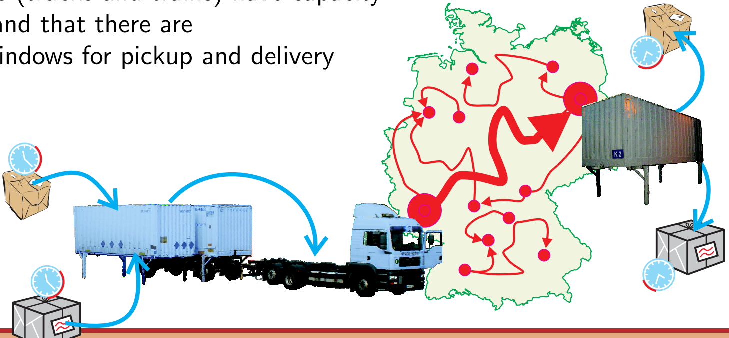
- Build a system which tells a logistics company what it needs to do to fulfill all transportation orders at minimum costs. [1–5]
  1. Find routes on the map and assignments of orders to containers and containers to trucks/trains which minimize the undelivered orders and the total distance for...
  2. multiple depots and pickup and delivery locations



- Build a system which tells a logistics company what it needs to do to fulfill all transportation orders at minimum costs. [1–5]
  1. Find routes on the map and assignments of orders to containers and containers to trucks/trains which minimize the undelivered orders and the total distance for…
  2. multiple depots and pickup and delivery locations, while considering that
  3. vehicles (trucks and trains) have capacity limits

Transportation Planning: Task

- Build a system which tells a logistics company what it needs to do to fulfill all transportation orders at minimum costs. [1–5]
  1. Find routes on the map and assignments of orders to containers and containers to trucks/trains which minimize the undelivered orders and the total distance for...
  2. multiple depots and pickup and delivery locations, while considering that
  3. vehicles (trucks and trains) have capacity limits and that there are
  4. time windows for pickup and delivery



Transportation Planning: Task

- Build a system which tells a logistics company what it needs to do to fulfill all transportation orders at minimum costs. [1–5]
  1. Find routes on the map and assignments of orders to containers and containers to trucks/trains which minimize the undelivered orders and the total distance for...
  2. multiple depots and pickup and delivery locations, while considering that
  3. vehicles (trucks and trains) have capacity limits and that there are
  4. time windows for pickup and delivery
  5. and constraints and laws.
  6. Time limit for optimization: 1 day

Transportation Planning: Problem

- Problem is complicated

Transportation Planning: Problem

- Problem is complicated
- No algorithm or existing solution available

Transportation Planning: Problem

- Problem is complicated
- No algorithm or existing solution available
- Problems like this one are also often $\mathcal{NP}$-hard

Transportation Planning: Problem

- Problem is complicated
- No algorithm or existing solution available
- Problems like this one are also often $\mathcal{NP}$-hard $\Rightarrow$ worst-case runtime needed to find the best possible solution grows like 2^n with number n of orders/cars

Transportation Planning: Problem

- Problem is complicated
- No algorithm or existing solution available
- Problems like this one are also often $\mathcal{NP}$-hard $\Rightarrow$ worst-case runtime needed to find the best possible solution grows like 2^n with number n of orders/cars $\Rightarrow$ probably not possible to develop a useful exact algorithm

Transportation Planning: Problem

- Problem is complicated
- No algorithm or existing solution available
- Problems like this one are also often $\mathcal{NP}$-hard $\Rightarrow$ worst-case runtime needed to find the best possible solution grows like 2^n with number n of orders/cars $\Rightarrow$ probably not possible to develop a useful exact algorithm
- Programming experience, data structure classes, Concrete Maths...
$\Rightarrow$ these alone cannot solve this issue (since $\mathcal{NP}$-hard)

Transportation Planning: Problem

- Problem is complicated
- No algorithm or existing solution available
- Problems like this one are also often $\mathcal{NP}$-hard $\Rightarrow$ worst-case runtime needed to find the best possible solution grows like 2^n with number n of orders/cars $\Rightarrow$ probably not possible to develop a useful exact algorithm
- Programming experience, data structure classes, Concrete Maths…
  $\Rightarrow$ these alone cannot solve this issue (since $\mathcal{NP}$-hard)
- Solution: adapt optimization algorithm (in our case: an Evolutionary Algorithm) to our problem [1–5]

Transportation Planning: Problem

- Problem is complicated
- No algorithm or existing solution available
- Problems like this one are also often $\mathcal{NP}$-hard $\Rightarrow$ worst-case runtime needed to find the best possible solution grows like 2^n with number n of orders/cars $\Rightarrow$ probably not possible to develop a useful exact algorithm
- Programming experience, data structure classes, Concrete Maths...
  $\Rightarrow$ these alone cannot solve this issue (since $\mathcal{NP}$-hard)
- Solution: adapt optimization algorithm (in our case: an Evolutionary Algorithm) to our problem [1–5]
- Before the problem was solved by hand, by manual planning with Excel sheets...

Transportation Planning: Problem

- Problem is complicated
- No algorithm or existing solution available
- Problems like this one are also often $\mathcal{NP}$-hard $\Rightarrow$ worst-case runtime needed to find the best possible solution grows like 2^n with number n of orders/cars $\Rightarrow$ probably not possible to develop a useful exact algorithm
- Programming experience, data structure classes, Concrete Maths... $\Rightarrow$ these alone cannot solve this issue (since $\mathcal{NP}$-hard)
- Solution: adapt optimization algorithm (in our case: an Evolutionary Algorithm) to our problem [1–5]
- Before the problem was solved by hand, by manual planning with Excel sheets...
- With an optimization algorithm, we can get better solutions than that.

Transportation Planning: Problem

- Problem is complicated
- No algorithm or existing solution available
- Problems like this one are also often $\mathcal{NP}$-hard $\Rightarrow$ worst-case runtime needed to find the best possible solution grows like 2^n with number n of orders/cars $\Rightarrow$ probably not possible to develop a useful exact algorithm
- Programming experience, data structure classes, Concrete Maths... $\Rightarrow$ these alone cannot solve this issue (since $\mathcal{NP}$-hard)
- Solution: adapt optimization algorithm (in our case: an Evolutionary Algorithm) to our problem [1–5]
- Before the problem was solved by hand, by manual planning with Excel sheets...
- With an optimization algorithm, we can get better solutions than that.
- In this course, you will learn how we can do such a thing.

Section Outline

What is optimization?

- What is optimization?

- What is optimization? [6]

- What is optimization? [6]

Definition (Optimization)

Optimization is the process of solving an optimization problem, i.e., finding suitable solutions for it.

- What is optimization? [6]
- Ok, so what is an optimization problem?

Definition (Optimization Problem: Economical View)

An optimization problem is a situation which requires deciding for one choice from a set of possible alternatives in order to reach a predefined/required benefit at minimal costs.

- What is optimization? [6]
- Ok, so what is an optimization problem?

Definition (Optimization Problem: Economical View)

An optimization problem is a situation which requires deciding for one choice from a set of possible alternatives in order to reach a predefined/required benefit at minimal costs.

Definition (Optimization Problem: Simplified Mathematical View)

Solving an optimization problem requires finding an input value $x^{\star}$ for which a mathematical function f takes on the smallest possible value (while usually obeying to some restrictions on the possible values of $x^{\star}$).

What is optimization?

- What is optimization? [6]
- Ok, so what is an optimization problem?
- Some optimization problems are too complicated to find the perfect (optimal) solution (in reasonable time)...

- What is optimization? [6]
- Ok, so what is an optimization problem?
- Some optimization problems are too complicated to find the perfect (optimal) solution (in reasonable time)...
- Then, we want to find approximate solutions: solutions which are "as good as possible" within a feasible time

- What is optimization? [6]
- Ok, so what is an optimization problem?
- Some optimization problems are too complicated to find the perfect (optimal) solution (in reasonable time)...
- Then, we want to find approximate solutions: solutions which are "as good as possible" within a feasible time
- This is what metaheuristic optimization algorithms do.

What is optimization?

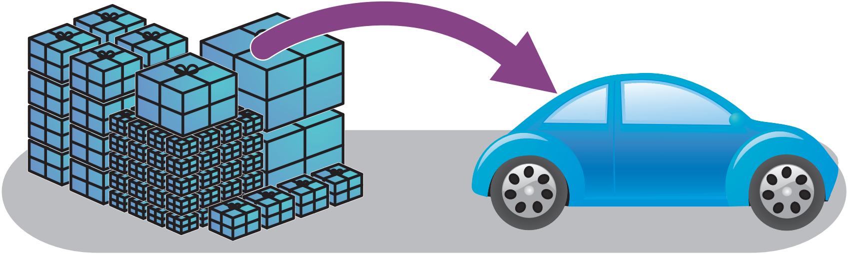
- What is optimization? [6]
- Ok, so what is an optimization problem?
- Some optimization problems are too complicated to find the perfect (optimal) solution (in reasonable time)...
- Then, we want to find approximate solutions: solutions which are "as good as possible" within a feasible time
- This is what metaheuristic optimization algorithms do. This is what we will learn in this course.

Optimization

- Many questions in the real world are *optimization problems*

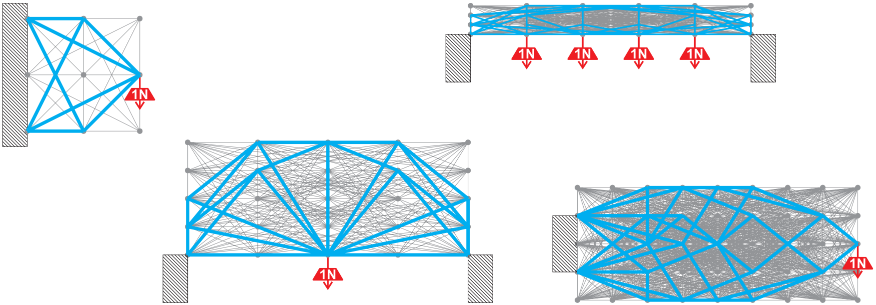
- Many questions in the real world are *optimization problems*, e.g.,
  - Find the *shortest* tour for a salesman to visit a certain set of cities in China and return to Hefei!

- Many questions in the real world are *optimization problems*, e.g.,
  - Find the *shortest* tour for a salesman to visit a certain set of cities
  - I need to transport n items from here to another city 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?

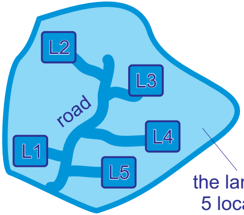


Optimization

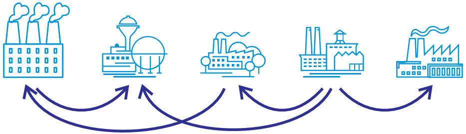
- Many questions in the real world are *optimization problems*, e.g.,
  - Find the *shortest* tour for a salesman to visit a certain set of cities
  - I need to transport n items from here to another city
  - How can I construct a truss which can hold a certain weight with at most a certain amount of iron?



- Many questions in the real world are *optimization problems*, e.g.,
  - Find the *shortest* tour for a salesman to visit a certain set of cities
  - I need to transport n items from here to another city
  - Construct a truss which can hold a certain weight
  - 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*.

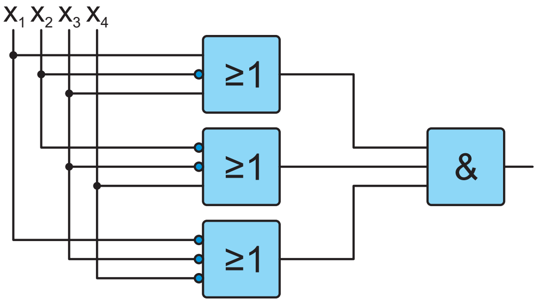


the land with
5 locations



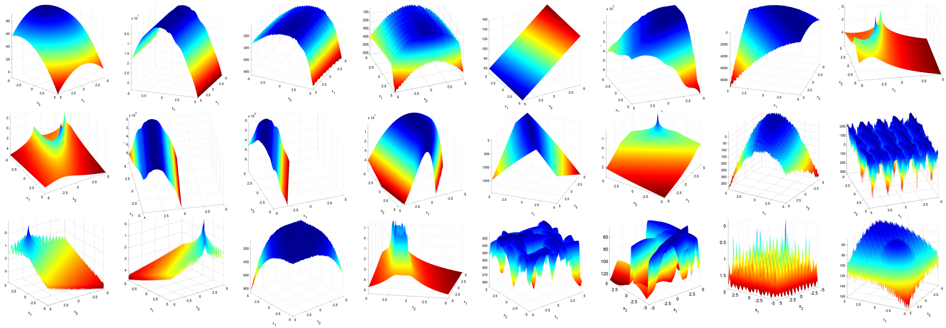
5 workshops and goods flows betwee them
which need to be assigned to locations

- Many questions in the real world are *optimization problems*, e.g.,
  - Find the *shortest* tour for a salesman to visit a certain set of cities
  - I need to transport n items from here to another city
  - Construct a truss which can hold a certain weight
  - Assign workshops to locations
  - 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)?



Optimization

- Many questions in the real world are *optimization problems*, e.g.,
  - Find the *shortest* tour for a salesman to visit a certain set of cities
  - I need to transport n items from here to another city
  - Construct a truss which can hold a certain weight
  - Assign workshops to locations
  - Satisfy Boolean formula
  - Find the minima of complex, multi-dimensional mathematical formulas

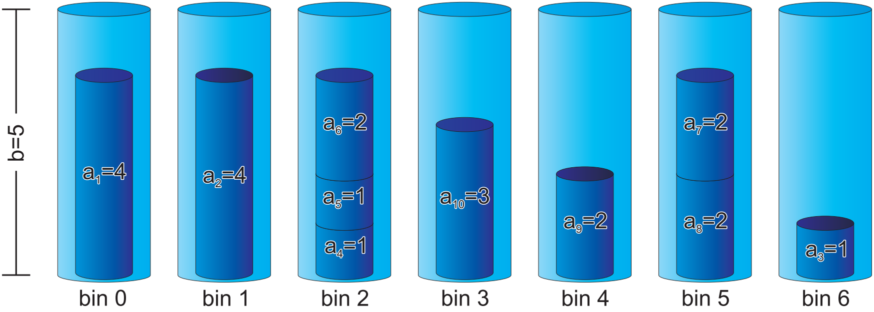


More Examples for Optimization Problems

- Let us look at some more examples for optimization (problems)

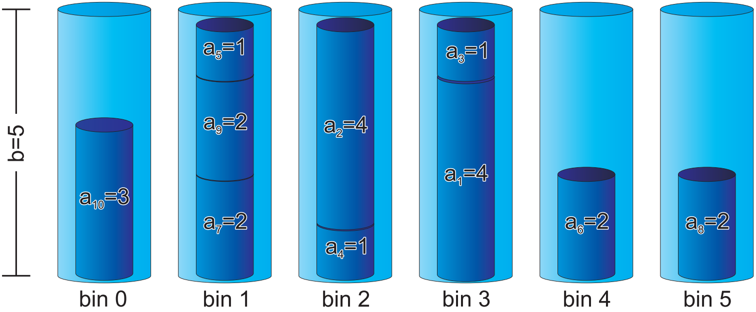
- Let us look at some more examples for optimization (problems)

1. Can we pack $n = 10$ objects of sizes $a_1, a_2, \dots, a_{10}$ into bins of size $b = 5$? [7] (or What is the minimum number k of bins that we need?)



- Let us look at some more examples for optimization (problems)

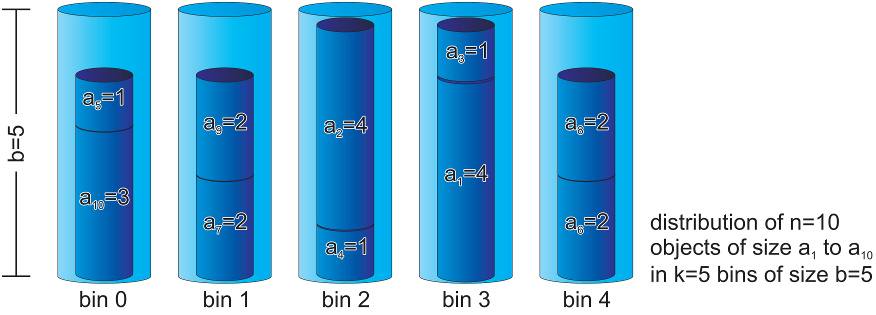
  1. Can we pack $n = 10$ objects of sizes $a_1, a_2, \dots, a_{10}$ into bins of size $b = 5$? [7] (or What is the minimum number k of bins that we need?)



More Examples for Optimization Problems

- Let us look at some more examples for optimization (problems)

  1. Can we pack $n = 10$ objects of sizes $a_1, a_2, \dots, a_{10}$ into bins of size $b = 5$? [7] (or What is the minimum number k of bins that we need?)

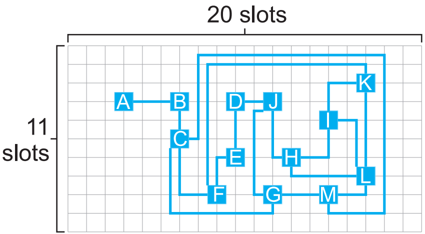


More Examples for Optimization Problems

- Let us look at some more examples for optimization (problems)
  1. Can we pack $n = 10$ objects of sizes $a_1, a_2, \dots, a_{10}$ into bins of size $b = 5$? [7] (or What is the minimum number k of bins that we need? ... Actually, this is exactly the same problem as the one with the car and moving to a new flat before!)

- Let us look at some more examples for optimization (problems)

  1. Can we pack $n = 10$ objects of sizes $a_1, a_2, \dots, a_{10}$ into bins of size $b = 5$? [7]
  2. How to place chips on a circuit board while minimizing wire length? [8, 9]



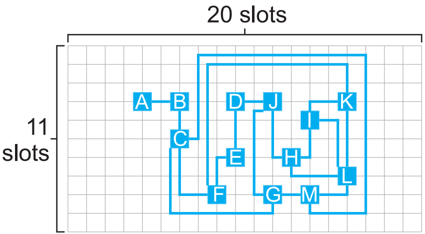
S=20×11

C={A, B, C, D, E, F, G, H, I, J, K, L, M}

R={(A,B), (B,C), (C,F), (C,G), (C,M), (D,E), (D,J), (E,F), (F,K), (G,J), (G,M), (H,J), (H,I), (H,L), (I,K), (I,L), (K,L), (L,M) }

More Examples for Optimization Problems

- Let us look at some more examples for optimization (problems)
  1. Can we pack $n = 10$ objects of sizes $a_1, a_2, \dots, a_{10}$ into bins of size $b = 5$? [7]
  2. How to place chips on a circuit board while minimizing wire length? [8, 9]



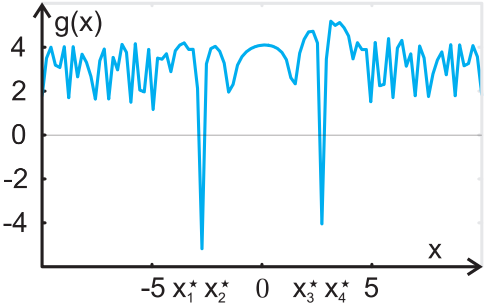
S=20×11

C={A, B, C, D, E, F, G, H, I, J, K, L, M}

R={(A,B), (B,C), (C,F), (C,G), (C,M), (D,E), (D,J), (E,F), (F,K), (G,J), (G,M), (H,J), (H,I), (H,L), (I,K), (I,L), (K,L), (L,M) }

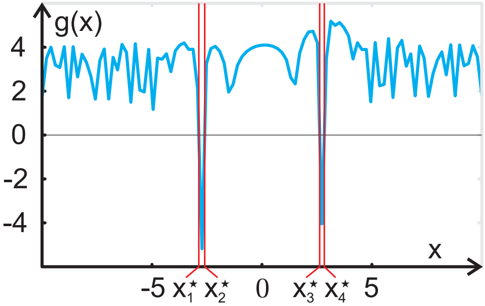
- Let us look at some more examples for optimization (problems)

1. Can we pack $n = 10$ objects of sizes $a_1, a_2, \dots, a_{10}$ into bins of size $b = 5$? [7]
2. How to place chips on a circuit board while minimizing wire length? [8, 9]
3. Find the roots [10–13] $x_0 : g(x_0) = 0$ of the function $g(x) = 5 + \ln\left|\sin\left(x^2 + e^{\frac{x}{\tan x}}\right) + \cos x\right| - \tan^{-1}|x+3|$!



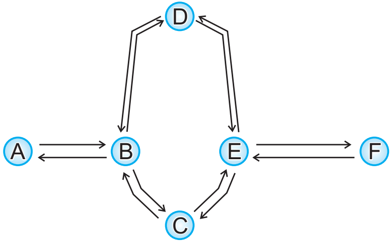
- Let us look at some more examples for optimization (problems)

1. Can we pack $n = 10$ objects of sizes $a_1, a_2, \dots, a_{10}$ into bins of size $b = 5$? [7]
2. How to place chips on a circuit board while minimizing wire length? [8, 9]
3. Find the roots [10–13] $x_0 : g(x_0) = 0$ of the function $g(x) = 5 + \ln\left|\sin\left(x^2 + e^{\frac{x}{\tan x}}\right) + \cos x\right| - \tan^{-1}|x+3|$!



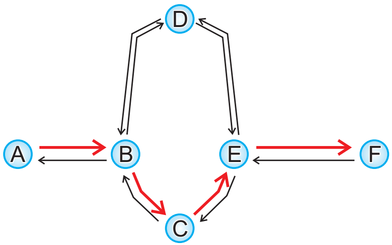
- Let us look at some more examples for optimization (problems)

1. Can we pack $n = 10$ objects of sizes $a_1, a_2, \dots, a_{10}$ into bins of size $b = 5$? [7]
2. How to place chips on a circuit board while minimizing wire length? [8, 9]
3. Find the roots [10–13] $x_0 : g(x_0) = 0$ of the function $g(x) = 5 + \ln\left|\sin\left(x^2 + e^{\frac{x}{\tan x}}\right) + \cos x\right| - \tan^{-1}|x+3|$!
4. Find the shortest path between two nodes in a network! [14–16]



- Let us look at some more examples for optimization (problems)

  1. Can we pack $n = 10$ objects of sizes $a_1, a_2, \dots, a_{10}$ into bins of size $b = 5$? [7]
  2. How to place chips on a circuit board while minimizing wire length? [8, 9]
  3. Find the roots [10–13] $x_0 : g(x_0) = 0$ of the function $g(x) = 5 + \ln\left|\sin\left(x^2 + e^{\frac{x}{\tan x}}\right) + \cos x\right| - \tan^{-1}|x+3|$!
  4. Find the shortest path between two nodes in a network! [14–16]



- Let us look at some more examples for optimization (problems)

  1. Can we pack $n = 10$ objects of sizes $a_1, a_2, \dots, a_{10}$ into bins of size $b = 5$? [7]
  2. How to place chips on a circuit board while minimizing wire length? [8, 9]
  3. Find the roots [10–13] $x_0 : g(x_0) = 0$ of the function $g(x) = 5 + \ln \left| \sin\left(x^2 + e^{\frac{x}{\tan x}}\right) + \cos x \right| - \tan^{-1} |x + 3|$!
  4. Find the shortest path between two nodes in a network! [14–16]
  5. Assign sub-jobs $j_i \in J$ of jobs J to machines $m \in M$ under order and deadline constraints! [17–19]

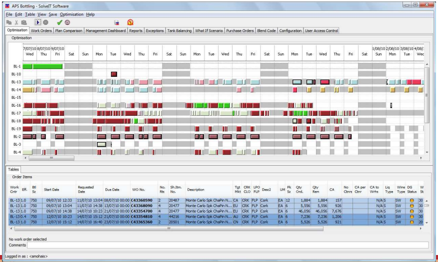
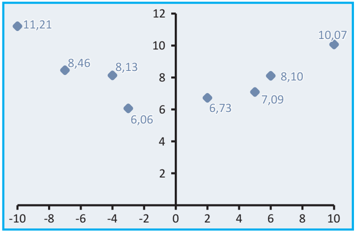


image source: [17]

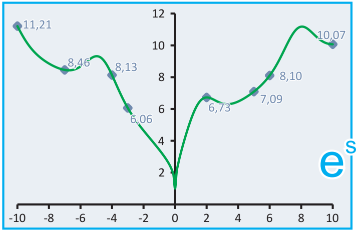
- Let us look at some more examples for optimization (problems)



6 Find a formula which fits to given measurements! [18, 20, 21]

More Examples for Optimization Problems

- Let us look at some more examples for optimization (problems)



$$e^{\sin x} + 3\sqrt{|x|}$$

6. Find a formula which fits to given measurements! [18, 20, 21]

More Examples for Optimization Problems

- Let us look at some more examples for optimization (problems)

6. Find a formula which fits to given measurements! [18, 20, 21]
7. Based on their past prices, which stocks are promising? [22–25]

More Examples for Optimization Problems

- Let us look at some more examples for optimization (problems)

6. Find a formula which fits to given measurements! [18, 20, 21]
7. Based on their past prices, which stocks are promising? [22–25]
8. Build an automatic way to detect MS lesions from MRI data [26] or cancer (risk) from patient data. [27]

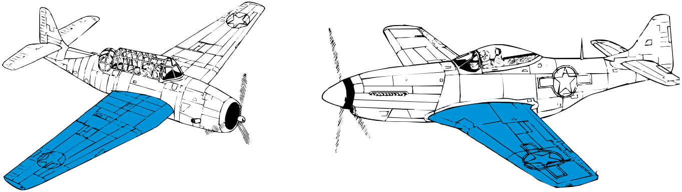
More Examples for Optimization Problems

- Let us look at some more examples for optimization (problems)

image source: [26]

6. Find a formula which fits to given measurements! [18, 20, 21]
7. Based on their past prices, which stocks are promising? [22–25]
8. Build an automatic way to detect MS lesions from MRI data [26] or cancer (risk) from patient data. [27]
9. Design airplane wings with minimum aerodynamic drag and maximum stability. [28–31]

- Let us look at some more examples for optimization (problems)



6. Find a formula which fits to given measurements! [18, 20, 21]
7. Based on their past prices, which stocks are promising? [22–25]
8. Build an automatic way to detect MS lesions from MRI data [26] or cancer (risk) from patient data. [27]
9. Design airplane wings with minimum aerodynamic drag and maximum stability. [28–31]
10. Select courses for your schedule maximizing both your expected score and practical utility for a future job while minimizing work requirements

Classification of Problem Types

- Are there different classes of optimization problems?

Classification of Problem Types

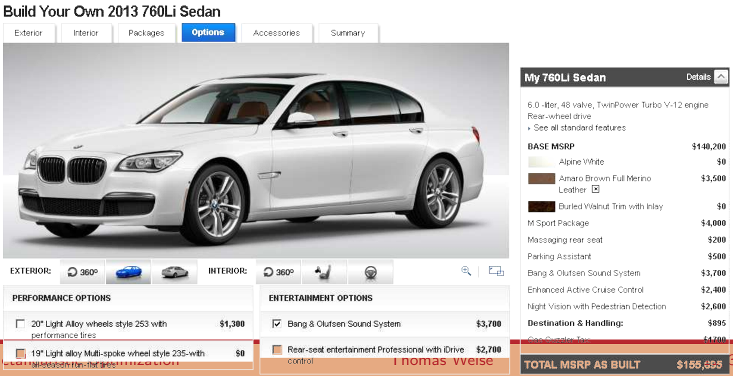
- Many problems are either combinatorial or numerical:

Classification of Problem Types

- Many problems are either combinatorial or numerical:

Definition (Combinatorial Optimization Problem)

Combinatorial optimization problems [32–38] are problems which have finitely many and discrete solutions.

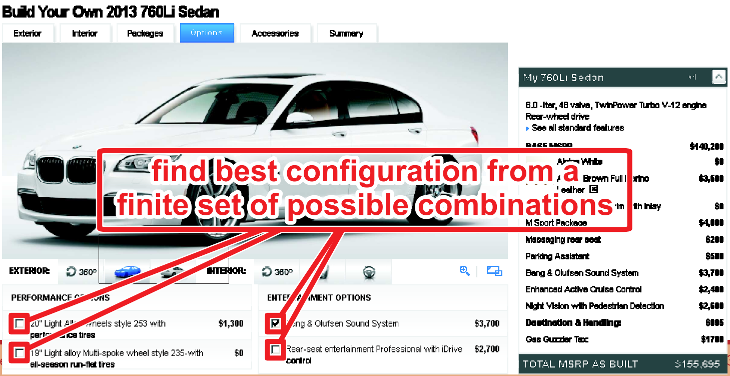


Classification of Problem Types

- Many problems are either combinatorial or numerical:

Definition (Combinatorial Optimization Problem)

Combinatorial optimization problems [32–38] are problems which have finitely many and discrete solutions.



- Many problems are either combinatorial or numerical:

Definition (Combinatorial Optimization Problem)

Combinatorial optimization problems [32–38] are problems which have finitely many and discrete solutions.

- Many problems are either combinatorial or numerical:

Definition (Combinatorial Optimization Problem)

Combinatorial optimization problems [32–38] are problems which have finitely many and discrete solutions.

Find the visiting sequence of cities corresponding to the shortest round-trip tour.

- Many problems are either combinatorial or numerical:

Definition (Combinatorial Optimization Problem)

Combinatorial optimization problems [32–38] are problems which have finitely many and discrete solutions,e.g.,

1. bit strings (i.e., many yes-no decisions) or
2. permutation of elements (order something).

- Many problems are either combinatorial or numerical:

Definition (Numerical Optimization Problem)

Numerical optimization problems are problems that are defined over numerical domains or involve real-valued decision variables. [10, 11, 40–43]

image source: [44]

- Many problems are either combinatorial or numerical:

Definition (Numerical Optimization Problem)

Numerical optimization problems are problems that are defined over numerical domains or involve real-valued decision variables. [10, 11, 40–43]

image source: [44]

Classification of Problem Types

- Many problems are either combinatorial or numerical:

Definition (Numerical Optimization Problem)

Numerical optimization problems are problems that are defined over numerical domains or involve real-valued decision variables. [10, 11, 40–43]

image source: [45]

- Many problems are either combinatorial or numerical:

Definition (Numerical Optimization Problem)

Numerical optimization problems are problems that are defined over numerical domains or involve real-valued decision variables. [10, 11, 40–43]

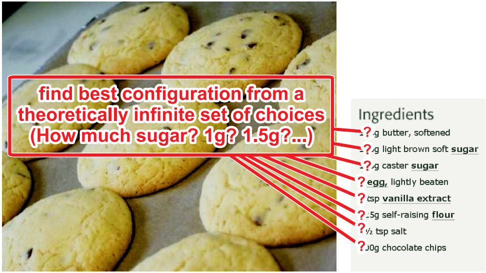


image source: [45]

Problem Types: Examples

- Are these problems combinatorial or numerical?

1) Traveling Salesman
2) Bin Packing
3) Truss Optimization
4) Quadratic Assignment
5) MAX-SAT
6) Minimize Function
7) Circuit Layout
8) Job Shop Scheduling
9) Routing
10) Find the roots of $g(x)$
11) Find mathematical formula
12) Stock Prediction
13) Medical Classification
14) Airplane Wing Design
15) Course Selection

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	
3)	Truss Optimization	
4)	Quadratic Assignment	
5)	MAX-SAT	
6)	Minimize Function	
7)	Circuit Layout	
8)	Job Shop Scheduling	
9)	Routing	
10)	Find the roots of $g(x)$	
11)	Find mathematical formula	
12)	Stock Prediction	
13)	Medical Classification	
14)	Airplane Wing Design	
15)	Course Selection	

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	combinatorial
3)	Truss Optimization	
4)	Quadratic Assignment	
5)	MAX-SAT	
6)	Minimize Function	
7)	Circuit Layout	
8)	Job Shop Scheduling	
9)	Routing	
10)	Find the roots of $g(x)$	
11)	Find mathematical formula	
12)	Stock Prediction	
13)	Medical Classification	
14)	Airplane Wing Design	
15)	Course Selection	

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	combinatorial
3)	Truss Optimization	theory: numerical
4)	Quadratic Assignment	
5)	MAX-SAT	
6)	Minimize Function	
7)	Circuit Layout	
8)	Job Shop Scheduling	
9)	Routing	
10)	Find the roots of $g(x)$	
11)	Find mathematical formula	
12)	Stock Prediction	
13)	Medical Classification	
14)	Airplane Wing Design	
15)	Course Selection	

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	combinatorial
3)	Truss Optimization	practice: maybe combinatorial
4)	Quadratic Assignment	
5)	MAX-SAT	
6)	Minimize Function	
7)	Circuit Layout	
8)	Job Shop Scheduling	
9)	Routing	
10)	Find the roots of $g(x)$	
11)	Find mathematical formula	
12)	Stock Prediction	
13)	Medical Classification	
14)	Airplane Wing Design	
15)	Course Selection	

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	combinatorial
3)	Truss Optimization	could be either
4)	Quadratic Assignment	
5)	MAX-SAT	
6)	Minimize Function	
7)	Circuit Layout	
8)	Job Shop Scheduling	
9)	Routing	
10)	Find the roots of $g(x)$	
11)	Find mathematical formula	
12)	Stock Prediction	
13)	Medical Classification	
14)	Airplane Wing Design	
15)	Course Selection	

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	combinatorial
3)	Truss Optimization	could be either
4)	Quadratic Assignment	combinatorial
5)	MAX-SAT	
6)	Minimize Function	
7)	Circuit Layout	
8)	Job Shop Scheduling	
9)	Routing	
10)	Find the roots of $g(x)$	
11)	Find mathematical formula	
12)	Stock Prediction	
13)	Medical Classification	
14)	Airplane Wing Design	
15)	Course Selection	

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	combinatorial
3)	Truss Optimization	could be either
4)	Quadratic Assignment	combinatorial
5)	MAX-SAT	combinatorial
6)	Minimize Function	
7)	Circuit Layout	
8)	Job Shop Scheduling	
9)	Routing	
10)	Find the roots of $g(x)$	
11)	Find mathematical formula	
12)	Stock Prediction	
13)	Medical Classification	
14)	Airplane Wing Design	
15)	Course Selection	

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	combinatorial
3)	Truss Optimization	could be either
4)	Quadratic Assignment	combinatorial
5)	MAX-SAT	combinatorial
6)	Minimize Function	numerical
7)	Circuit Layout	
8)	Job Shop Scheduling	
9)	Routing	
10)	Find the roots of $g(x)$	
11)	Find mathematical formula	
12)	Stock Prediction	
13)	Medical Classification	
14)	Airplane Wing Design	
15)	Course Selection	

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	combinatorial
3)	Truss Optimization	could be either
4)	Quadratic Assignment	combinatorial
5)	MAX-SAT	combinatorial
6)	Minimize Function	numerical
7)	Circuit Layout	combinatorial
8)	Job Shop Scheduling	
9)	Routing	
10)	Find the roots of $g(x)$	
11)	Find mathematical formula	
12)	Stock Prediction	
13)	Medical Classification	
14)	Airplane Wing Design	
15)	Course Selection	

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	combinatorial
3)	Truss Optimization	could be either
4)	Quadratic Assignment	combinatorial
5)	MAX-SAT	combinatorial
6)	Minimize Function	numerical
7)	Circuit Layout	combinatorial
8)	Job Shop Scheduling	combinatorial
9)	Routing	
10)	Find the roots of $g(x)$	
11)	Find mathematical formula	
12)	Stock Prediction	
13)	Medical Classification	
14)	Airplane Wing Design	
15)	Course Selection	

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	combinatorial
3)	Truss Optimization	could be either
4)	Quadratic Assignment	combinatorial
5)	MAX-SAT	combinatorial
6)	Minimize Function	numerical
7)	Circuit Layout	combinatorial
8)	Job Shop Scheduling	combinatorial
9)	Routing	combinatorial
10)	Find the roots of $g(x)$	
11)	Find mathematical formula	
12)	Stock Prediction	
13)	Medical Classification	
14)	Airplane Wing Design	
15)	Course Selection	

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	combinatorial
3)	Truss Optimization	could be either
4)	Quadratic Assignment	combinatorial
5)	MAX-SAT	combinatorial
6)	Minimize Function	numerical
7)	Circuit Layout	combinatorial
8)	Job Shop Scheduling	combinatorial
9)	Routing	combinatorial
10)	Find the roots of $g(x)$	numerical
11)	Find mathematical formula	
12)	Stock Prediction	
13)	Medical Classification	
14)	Airplane Wing Design	
15)	Course Selection	

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	combinatorial
3)	Truss Optimization	could be either
4)	Quadratic Assignment	combinatorial
5)	MAX-SAT	combinatorial
6)	Minimize Function	numerical
7)	Circuit Layout	combinatorial
8)	Job Shop Scheduling	combinatorial
9)	Routing	combinatorial
10)	Find the roots of $g(x)$	numerical
11)	Find mathematical formula	(usually need real numbers)
12)	Stock Prediction	
13)	Medical Classification	
14)	Airplane Wing Design	
15)	Course Selection	

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	combinatorial
3)	Truss Optimization	could be either
4)	Quadratic Assignment	combinatorial
5)	MAX-SAT	combinatorial
6)	Minimize Function	numerical
7)	Circuit Layout	combinatorial
8)	Job Shop Scheduling	combinatorial
9)	Routing	combinatorial
10)	Find the roots of $g(x)$	numerical
11)	Find mathematical formula	(usually need real numbers)
12)	Stock Prediction	(usually need real numbers)
13)	Medical Classification	
14)	Airplane Wing Design	
15)	Course Selection	

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	combinatorial
3)	Truss Optimization	could be either
4)	Quadratic Assignment	combinatorial
5)	MAX-SAT	combinatorial
6)	Minimize Function	numerical
7)	Circuit Layout	combinatorial
8)	Job Shop Scheduling	combinatorial
9)	Routing	combinatorial
10)	Find the roots of $g(x)$	numerical
11)	Find mathematical formula	(usually need real numbers)
12)	Stock Prediction	(usually need real numbers)
13)	Medical Classification	(usually need real numbers)
14)	Airplane Wing Design	
15)	Course Selection	

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	combinatorial
3)	Truss Optimization	could be either
4)	Quadratic Assignment	combinatorial
5)	MAX-SAT	combinatorial
6)	Minimize Function	numerical
7)	Circuit Layout	combinatorial
8)	Job Shop Scheduling	combinatorial
9)	Routing	combinatorial
10)	Find the roots of $g(x)$	numerical
11)	Find mathematical formula	(usually need real numbers)
12)	Stock Prediction	(usually need real numbers)
13)	Medical Classification	(usually need real numbers)
14)	Airplane Wing Design	numerical
15)	Course Selection	

Problem Types: Examples

- Are these problems combinatorial or numerical?

1)	Traveling Salesman	combinatorial
2)	Bin Packing	combinatorial
3)	Truss Optimization	could be either
4)	Quadratic Assignment	combinatorial
5)	MAX-SAT	combinatorial
6)	Minimize Function	numerical
7)	Circuit Layout	combinatorial
8)	Job Shop Scheduling	combinatorial
9)	Routing	combinatorial
10)	Find the roots of $g(x)$	numerical
11)	Find mathematical formula	(usually need real numbers)
12)	Stock Prediction	(usually need real numbers)
13)	Medical Classification	(usually need real numbers)
14)	Airplane Wing Design	numerical
15)	Course Selection	combinatorial

Section Outline

- What is an optimization algorithm?

What is an optimization algorithm?

- What is an optimization algorithm?

Definition (Algorithm)

An algorithm is a finite set of well-defined instructions for accomplishing some task. It starts in some initial state and usually terminates in a final state.

What is an optimization algorithm?

- What is an optimization algorithm?

Definition (Algorithm)

An algorithm is a finite set of well-defined instructions for accomplishing some task. It starts in some initial state and usually terminates in a final state.

Algorithms are the very basic of computer science. An algorithm tells us what we can do to solve a given task.

- What is an optimization algorithm?

Definition (Algorithm)

An algorithm is a finite set of well-defined instructions for accomplishing some task. It starts in some initial state and usually terminates in a final state.

Definition (Optimization Algorithm)

An optimization algorithm is an algorithm suitable to solve optimization problems.

- What is an optimization algorithm?

Definition (Algorithm)

An algorithm is a finite set of well-defined instructions for accomplishing some task. It starts in some initial state and usually terminates in a final state.

Definition (Optimization Algorithm)

An optimization algorithm is an algorithm suitable to solve optimization problems.

Optimization algorithms tell us how to find solutions which are *rated best* (or at least well) from a set of possible solutions, for a general class of problems.

Section Outline

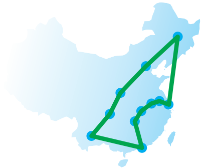
Heuristic Optimization

- In optimization, there exist *exact* and *heuristic* algorithms.

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.

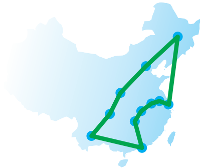


- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.
  - Theory proofs that the time to find this tour may grow exponentially with the number of cities we want to visit.

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
    - Clearly, there is (at least) one shortest tour.
    - Finding the best tour, i.e., exact algorithms, may take too long.

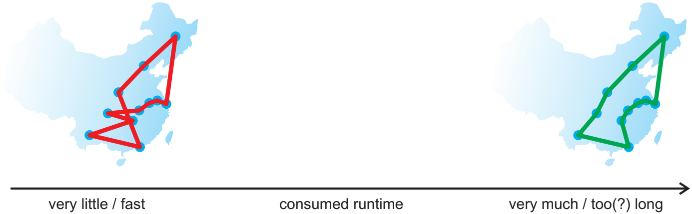
consumed runtime to find tour: very much / too(?) long

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.
  - Finding the best tour, i.e., exact algorithms, may take too long.
  - But we can find just *some* tour very quickly.



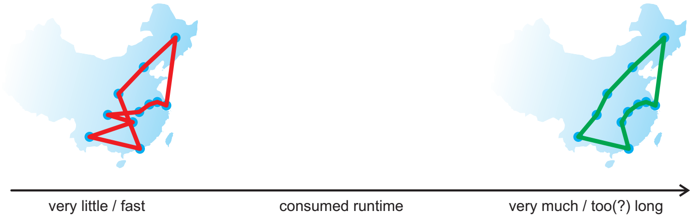
consumed runtime to find tour: very much / too(?) long

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
    - Clearly, there is (at least) one shortest tour.
    - Finding the best tour, i.e., exact algorithms, may take too long.
    - But we can find just *some* tour very quickly.



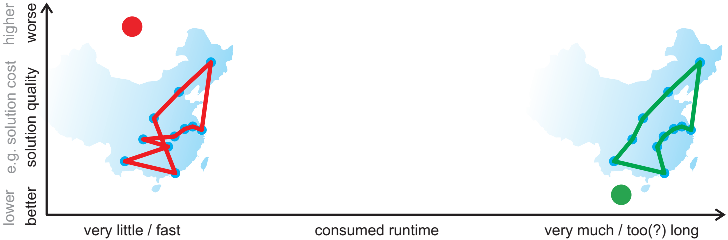
Heuristic Optimization

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.
  - Finding the best tour, i.e., exact algorithms, may take too long.
  - But we can find just *some* tour very quickly.
  - Of course the quality of that tour will be lower: the tour will be longer than the best one.



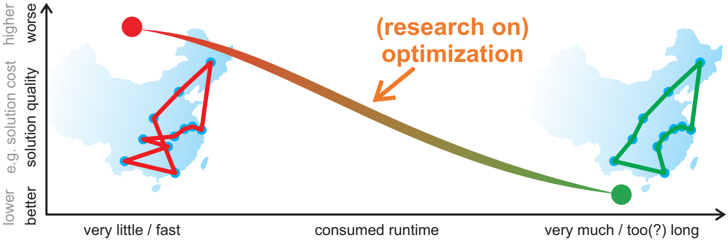
Heuristic Optimization

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.
  - Finding the best tour, i.e., exact algorithms, may take too long.
  - But we can find just *some* tour very quickly.
  - Of course the quality of that tour will be lower: the tour will be longer than the best one.



Heuristic Optimization

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.
  - Finding the best tour, i.e., exact algorithms, may take too long.
  - But we can find just *some* tour very quickly.
  - Of course the quality of that tour will be lower.
  - (Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible.



Heuristic Optimization

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.
  - Finding the best tour, i.e., exact algorithms, may take too long.
  - But we can find just *some* tour very quickly.
  - Of course the quality of that tour will be lower.
  - (Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible.
  - Optimization often means to trade-off between solution quality and runtime.

Heuristic Optimization

- In optimization, there exist *exact* and *heuristic* algorithms.
- Let's again look at the classical "Traveling Salesman Problem" (TSP).
  - Clearly, there is (at least) one shortest tour.
  - Finding the best tour, i.e., exact algorithms, may take too long.
  - But we can find just *some* tour very quickly.
  - Of course the quality of that tour will be lower.
  - (Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible.
  - Optimization often means to trade-off between solution quality and runtime and development time (the time from the definition of the problem until we have a software for producing solutions).

- Heuristics are often specialized algorithms: Say we have an approximate algorithm for the TSP, or one for the Bin Packing problem…

- Heuristics are often specialized algorithms: Say we have an approximate algorithm for the TSP, or one for the Bin Packing problem...
- However, there are may different optimization problems.

Metaheuristics

- Heuristics are often specialized algorithms: Say we have an approximate algorithm for the TSP, or one for the Bin Packing problem. . .
- However, there are may different optimization problems and often we won't see a "pure" TSP in practice.

Metaheuristics

- Heuristics are often specialized algorithms: Say we have an approximate algorithm for the TSP, or one for the Bin Packing problem...
- However, there are may different optimization problems and often we won't see a "pure" TSP in practice, there usually will be additional constraints and restrictions or multiple cars etc..

- Heuristics are often specialized algorithms: Say we have an approximate algorithm for the TSP, or one for the Bin Packing problem…
- However, there are may different optimization problems and often we won't see a "pure" TSP in practice, there usually will be additional constraints and restrictions or multiple cars etc. (see, for instance, the initial logistics scenario).

Metaheuristics

- Heuristics are often specialized algorithms: Say we have an approximate algorithm for the TSP, or one for the Bin Packing problem...
- However, there are may different optimization problems.
- Should we develop a completely new method for each problem?

Metaheuristics

- Heuristics are often specialized algorithms: Say we have an approximate algorithm for the TSP, or one for the Bin Packing problem...
- However, there are may different optimization problems.
- Should we develop a completely new method for each problem?
- No. We want general algorithms that can be adapted to different problems.

- Heuristics are often specialized algorithms: Say we have an approximate algorithm for the TSP, or one for the Bin Packing problem...
- However, there are may different optimization problems.
- Should we develop a completely new method for each problem?
- No. We want general algorithms that can be adapted to different problems. (also to reduce the development time)

Definition (Metaheuristic)

A metaheuristic is a method for solving very general classes of problems. It combines objective functions or heuristics in an abstract and hopefully efficient way, usually by treating them as black box-procedures. [46, 47]

- Heuristics are often specialized algorithms: Say we have an approximate algorithm for the TSP, or one for the Bin Packing problem...
- However, there are may different optimization problems.
- Should we develop a completely new method for each problem?
- No. We want general algorithms that can be adapted to different problems. (also to reduce the development time ... we often want a prototype quickly and can add more complex logic later)

Definition (Metaheuristic)

A metaheuristic is a method for solving very general classes of problems. It combines objective functions or heuristics in an abstract and hopefully efficient way, usually by treating them as black box-procedures. [46, 47]

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

- Algorithm performance has two dimensions [48, 49]:

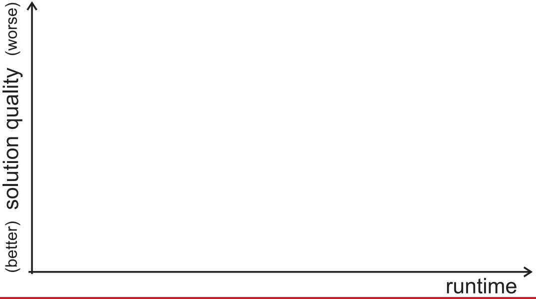
Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

- Algorithm performance has two dimensions [48, 49]: solution quality

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

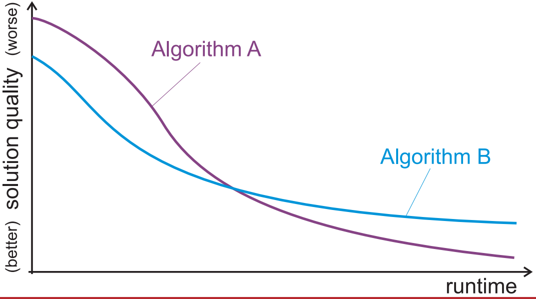
- Algorithm performance has two dimensions [48, 49]: solution quality and required runtime



Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

- Algorithm performance has two dimensions [48, 49]: solution quality and required runtime
- Anytime Algorithms [50] are optimization methods which maintain an approximate solution at *any time* during their run and iteratively improve this guess.



Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

- Algorithm performance has two dimensions [48, 49]: solution quality and required runtime
- Anytime Algorithms [50] are optimization methods which maintain an approximate solution at *any time* during their run and iteratively improve this guess.
- All metaheuristics are Anytime Algorithms.

Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

- Algorithm performance has two dimensions [48, 49]: solution quality and required runtime
- Anytime Algorithms [50] are optimization methods which maintain an approximate solution at *any time* during their run and iteratively improve this guess.
- All metaheuristics are Anytime Algorithms.
- Several exact methods like Branch-and-Bound [51–53] are Anytime Algorithms.

Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

- Algorithm performance has two dimensions [48, 49]: solution quality and required runtime
- Anytime Algorithms [50] are optimization methods which maintain an approximate solution at *any time* during their run and iteratively improve this guess.
- All metaheuristics are Anytime Algorithms.
- Several exact methods like Branch-and-Bound [51–53] are Anytime Algorithms.
- Consequence: Most optimization algorithms produce approximate solutions of different qualities at different points during their process.

Performance and Anytime Algorithms

"(Meta-)Heuristic optimization algorithms try to find solutions which are as good as possible as fast as possible."

- Algorithm performance has two dimensions [48, 49]: solution quality and required runtime
- Anytime Algorithms [50] are optimization methods which maintain an approximate solution at *any time* during their run and iteratively improve this guess.
- All metaheuristics are Anytime Algorithms.
- Several exact methods like Branch-and-Bound [51–53] are Anytime Algorithms.
- Consequence: Most optimization algorithms produce approximate solutions of different qualities at different points during their process.
- We can let them run arbitrarily long, there usually is no explicit, natural end point.

Timing Features

- Optimization problems and algorithms can be divided into online and offline processes.

- Optimization problems and algorithms can be divided into online and offline processes.

Definition (Online Optimization)

Online optimization problems are tasks that need to be solved quickly in a time span usually ranging between ten milliseconds to a few minutes.

- Optimization problems and algorithms can be divided into online and offline processes.

Definition (Online Optimization)

Online optimization problems are tasks that need to be solved quickly in a time span usually ranging between ten milliseconds to a few minutes.

Definition (Offline Optimization)

In offline optimization problems, time is not so important and a user is willing to wait maybe up to days or weeks for better results.

Timing Features

- Optimization problems and algorithms can be divided into online and offline processes.
- We will mainly deal with offline optimization.

Definition (Online Optimization)

Online optimization problems are tasks that need to be solved quickly in a time span usually ranging between ten milliseconds to a few minutes.

Definition (Offline Optimization)

In offline optimization problems, time is not so important and a user is willing to wait maybe up to days or weeks for better results.

1. Introductory Example
2. Optimization Problems
3. Optimization Algorithms
4. Runtime vs. Solution Quality
5. **Other Algorithm Features**
6. Summary

Deterministic and Randomized Algorithms

- Two types of algorithms: deterministic and randomized

- Two types of algorithms: deterministic and randomized

Definition (Deterministic Algorithm)

In each execution step of a deterministic algorithm, there exists at most one way to proceed. If no way to proceed exists, the algorithm has terminated.

Deterministic and Randomized Algorithms

- Two types of algorithms: deterministic and randomized

Definition (Deterministic Algorithm)

In each execution step of a deterministic algorithm, there exists at most one way to proceed. If no way to proceed exists, the algorithm has terminated.

Example:

- *Exhaustive Enumeration:* Test all possible solutions one by one and, thus, find the optimum.

Deterministic and Randomized Algorithms

- Two types of algorithms: deterministic and randomized

Definition (Deterministic Algorithm)

In each execution step of a deterministic algorithm, there exists at most one way to proceed. If no way to proceed exists, the algorithm has terminated.

Example:

- *Exhaustive Enumeration:* Test all possible solutions one by one and, thus, find the optimum.
- *Deterministic Heuristic:* Solve the bin packing problem by putting the largest remaining element into a bin until the bin is full, then take the next bin, until all objects are packed.

- Two types of algorithms: deterministic and randomized

Definition (Randomized Algorithm)

A randomized or probabilistic algorithm includes at least one instruction that acts on the basis of random numbers. In other words, a probabilistic algorithm violates the constraint of determinism. [43, 54–57]

Deterministic and Randomized Algorithms

- Two types of algorithms: deterministic and randomized

Definition (Randomized Algorithm)

A randomized or probabilistic algorithm includes at least one instruction that acts on the basis of random numbers. In other words, a probabilistic algorithm violates the constraint of determinism. [43, 54–57]

Examples:

- *Random Sampling:* Keep creating solutions randomly until the time is up, remember the best.

Deterministic and Randomized Algorithms

- Two types of algorithms: deterministic and randomized

Definition (Randomized Algorithm)

A randomized or probabilistic algorithm includes at least one instruction that acts on the basis of random numbers. In other words, a probabilistic algorithm violates the constraint of determinism. [43, 54–57]

Examples:

- *Random Sampling:* Keep creating solutions randomly until the time is up, remember the best.
- *Randomized Local Search:* Create a random solution, then try to improve it by modifying it a bit (randomly). If the new one is better, keep it. Modify again.

Deterministic and Randomized Algorithms

- Two types of algorithms: deterministic and randomized
- We will mainly deal with randomized algorithms.

- Two types of algorithms: deterministic and randomized
- We will mainly deal with randomized algorithms.
- Most interesting problems cannot be solved to optimality.

Deterministic and Randomized Algorithms

- Two types of algorithms: deterministic and randomized
- We will mainly deal with randomized algorithms.
- Most interesting problems cannot be solved to optimality.
- Deterministic heuristics are often simple, fast, and provide solutions with mediocre approximation quality

Deterministic and Randomized Algorithms

- Two types of algorithms: deterministic and randomized
- We will mainly deal with randomized algorithms.
- Most interesting problems cannot be solved to optimality.
- Deterministic heuristics are often simple, fast, and provide solutions with mediocre approximation quality
- Algorithms often need to make choices (Test all BMW with white color first??) from which we don't know which choice is good

Deterministic and Randomized Algorithms

- Two types of algorithms: deterministic and randomized
- We will mainly deal with randomized algorithms.
- Most interesting problems cannot be solved to optimality.
- Deterministic heuristics are often simple, fast, and provide solutions with mediocre approximation quality
- Algorithms often need to make choices (Test all BMW with white color first??) from which we don't know which choice is good $\Longrightarrow$ flip a coin (i.e., avoid to always make same (potentially bad) choice)

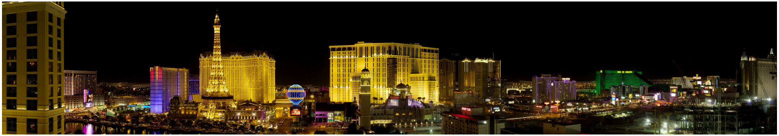
Deterministic and Randomized Algorithms

- Two types of algorithms: deterministic and randomized
- We will mainly deal with randomized algorithms.
- Most interesting problems cannot be solved to optimality.
- Deterministic heuristics are often simple, fast, and provide solutions with mediocre approximation quality
- Algorithms often need to make choices (Test all BMW with white color first??) from which we don't know which choice is good $\Longrightarrow$ flip a coin (i.e., avoid to always make same (potentially bad) choice) $\Longrightarrow$ randomize

Randomized Algorithms

- Two types of randomized algorithms: Las Vegas and Monte Carlo

- Two types of randomized algorithms: Las Vegas and Monte Carlo



Definition (Las Vegas Algorithm)

A Las Vegas Algorithm is a randomized algorithm that never returns a wrong result. It either returns the correct result, reports a failure, or does not return at all.

Randomized Algorithms

- Two types of randomized algorithms: Las Vegas and Monte Carlo

Definition (Las Vegas Algorithm)

A Las Vegas Algorithm is a randomized algorithm that either returns the correct result, reports a failure, or does not return.

Definition (Monte Carlo Algorithm)

A Monte Carlo Algorithm always terminates. Its result, however, can either be correct or incorrect. [58–61]

- Two types of randomized algorithms: Las Vegas and Monte Carlo
- We will mainly deal with Monte Carlo Algorithms.

Definition (Las Vegas Algorithm)

A Las Vegas Algorithm is a randomized algorithm that either returns the correct result, reports a failure, or does not return.

Definition (Monte Carlo Algorithm)

A Monte Carlo Algorithm always terminates. Its result, however, can either be correct or incorrect. [58–61]

- Two types of randomized algorithms: Las Vegas and Monte Carlo
- We will mainly deal with Monte Carlo Algorithms.
- ...since we often have a limited computational budget and need something that finishes within that time

Definition (Las Vegas Algorithm)

A Las Vegas Algorithm is a randomized algorithm that either returns the correct result, reports a failure, or does not return.

Definition (Monte Carlo Algorithm)

A Monte Carlo Algorithm always terminates. Its result, however, can either be correct or incorrect. [58–61]

Number of Objectives/Criteria

- Optimization problems and algorithms can be divided into single-objective and multi-objective optimization.

- Optimization problems and algorithms can be divided into single-objective and multi-objective optimization.

Definition (Single-Objective Optimization)

We have a single objective function f rating solution quality and want to find the solutions where f takes on minimal (or maximal) values.

- Optimization problems and algorithms can be divided into single-objective and multi-objective optimization.

Definition (Single-Objective Optimization)

We have a single objective function f rating solution quality and want to find the solutions where f takes on minimal (or maximal) values.

Definition (Multi-Objective Optimization)

We have a set $\vec{f}$ of different optimization criteria $f_i \in \vec{f}, i = 1 \dots n$ and want to find solutions which represent a good trade-off. [62–66]

- Optimization problems and algorithms can be divided into single-objective and multi-objective optimization.
- We will mainly deal with single-objective optimization but also discuss multi-objective methods.

Definition (Single-Objective Optimization)

We have a single objective function f rating solution quality and want to find the solutions where f takes on minimal (or maximal) values.

Definition (Multi-Objective Optimization)

We have a set $\vec{f}$ of different optimization criteria $f_i \in \vec{f}, i = 1 \dots n$ and want to find solutions which represent a good trade-off. [62–66]

- Optimization problems and algorithms can be divided into single-objective and multi-objective optimization.
- We will mainly deal with single-objective optimization but also discuss multi-objective methods and constraints.

Definition (Multi-Objective Optimization)

We have a set $\vec{f}$ of different optimization criteria $f_i \in \vec{f}, i = 1 \dots n$ and want to find solutions which represent a good trade-off. [62–66]

Definition (Constraint Optimization)

Single- or multi-objective Optimization where the solutions also need to fullful additional constraints. [67–69]

Section Outline

Summary

- Wide variety of optimization problem types
- Wide variety of optimization algorithms
- Wide variety of applications and requirements

Summary

- Wide variety of optimization problem types
- Wide variety of optimization algorithms
- Wide variety of applications and requirements
- One thing is clear: There is no single, perfect algorithm to solve all of them [70–73]

Summary

- Wide variety of optimization problem types
- Wide variety of optimization algorithms
- Wide variety of applications and requirements
- One thing is clear: There is no single, perfect algorithm to solve all of them [70–73]
- Experience is needed!
- We will try to get a good perspective on all of that
- ...and conduct many experiments in order to get experience

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, 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*, chapter 2, pages 29–53. Berlin/Heidelberg: Springer-Verlag, October 2009. doi: 10.1007/978-3-642-04039-9_2.
2. 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 (EvoWorkshops'09)*, volume 5484/2009 of *Lecture Notes in Computer Science (LNCS)*, pages 768–777, Tübingen, Germany: Eberhard-Karls-Universität Tübingen, Fakultät für Informations- und Kognitionswissenschaften, April 15–17, 2009. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/978-3-642-01129-0_87.
3. Thomas Weise, Alexander Podlich, Manfred Menze, and Christian Gorldt. Optimierte güterverkehrsplanung mit evolutionären algorithmen. *Industrie Management – Zeitschrift für industrielle Geschäftsprozesse*, 10(3):37–40, June 2, 2009. URL `http://www.logistics.de/logistik/branchen.nsf/D4FF2D93CA69B47FC12575CF004870C8/$File/gueterverkehrsplanung_optimierung_algorithmen_weise_podlich_menze_gorldt_gitopdf.pdf`.
4. Alexander Podlich. Intelligente planung und optimierung des güterverkehrs auf straße und schiene mit evolutionären algorithmen. Master's thesis, Kassel, Hesse, Germany: University of Kassel, Fachbereich 16: Elektrotechnik/Informatik, Distributed Systems Group, February 2009.
5. Manfred Menze. Evolutionäre algorithmen zur ad-hoc-tourenplanung: Inwest – intelligente wechselbrückensteuerung. Technical report, Kassel, Hesse, Germany: Micromata GmbH, November 1, 2010. URL `http://www.micromata.de/fileadmin/user_upload/Case_studies/cs_InWeSt.pdf`.
6. Alexander M. Bronstein and Michael M. Bronstein. Numerical optimization, 2008. URL `http://tosca.cs.technion.ac.il/book/slides/Milano08_optimization.ppt`.
7. Michael R. Garey and David Stifler Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Series of Books in the Mathematical Sciences. New York, NY, USA: W. H. Freeman and Company, 1979. ISBN 0-7167-1044-7, 0-7167-1045-5, 978-0-7167-1044-8, and 978-0-7167-1045-5. URL `http://books.google.de/books?id=mdBxHAAACAAJ`.

Bibliography II

8. 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.
9. Tatiana Kalganova and Julian Francis Miller. Evolving more efficient digital circuits by allowing circuit layout evolution and multi-objective fitness. In Adrian Stoica, Jason D. Lohn, and Didier Keymeulen, editors, *Evolvable Hardware – Proceedings of 1st NASA/DoD Workshop on Evolvable Hardware (EH'99)*, pages 54–63, Pasadena, CA, USA: Jet Propulsion Laboratory, California Institute of Technology (Caltech), June 19–21, 1999. Washington, DC, USA: IEEE Computer Society. doi: 10.1109/EH.1999.785435. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.47.948.
10. Xin Yao, Yong Liu, and Guangming Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, 3(2):82–102, July 1999. doi: 10.1109/4235.771163. URL http://www.u-aizu.ac.jp/~yliu/publication/tec22r2_online.ps.gz.
11. Thomas Bäck, Frank Hoffmeister, and Hans-Paul Schwefel. A survey of evolution strategies. In Richard K. Belew and Lashon Bernard Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA'91)*, pages 2–9, San Diego, CA, USA: University of California (UCSD), July 13–16, 1991. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. URL http://130.203.133.121:8080/viewdoc/summary?doi=10.1.1.42.3375.
12. 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.
13. Zbigniew Michalewicz. Genetic algorithms, numerical optimization, and constraints. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA'95)*, pages 151–158., Pittsburgh, PA, USA: University of Pittsburgh, July 15–19, 1995. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. URL http://www.cs.adelaide.edu.au/~zbyszek/Papers/p16.pdf.
14. Edsger Wybe Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. URL http://www-m3.ma.tum.de/twiki/pub/MN0506/WebHome/dijkstra.pdf.
15. Robert W Floyd. Algorithm 97 (shortest path). *Communications of the ACM (CACM)*, 5(6):345, June 1, 1962. doi: 10.1145/367766.368168.
16. Stephen Warshall. A theorem on boolean matrices. *Journal of the Association for Computing Machinery (JACM)*, 9(1): 11–12, January 1962. doi: 10.1145/321105.321107.

17. Arvind S. Mohais, Sven Schellenberg, Maksud Ibrahimov, Neal Wagner, and Zbigniew Michalewicz. An evolutionary approach to practical constraints in scheduling: A case-study of the wine bottling problem. In Raymond Chiong, Thomas Weise, and Zbigniew Michalewicz, editors, *Variants of Evolutionary Algorithms for Real-World Applications*, chapter 2, pages 31–58. Berlin/Heidelberg: Springer-Verlag, 2011. doi: 10.1007/978-3-642-23424-8_2.
18. 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.
19. Federico Della Croce, Roberto Tadei, and Giuseppe Volta. A genetic algorithm for the job shop problem. *Computers & Operations Research*, 22(1):15–24, January 1995. doi: 10.1016/0305-0548(93)E0015-L.
20. John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Bradford Books. Cambridge, MA, USA: MIT Press, December 1992. ISBN 0-262-11170-5 and 978-0-262-11170-6. URL http://books.google.de/books?id=Bhtxo60BV0EC. 1992 first edition, 1993 second edition.
21. Douglas A. Augusto and Helio Joseé Correa Barbosa. Symbolic regression via genetic programming. In Felipe M. G. França and Carlos H. C. Ribeiro, editors, *Proceedings of the VI Brazilian Symposium on Neural Networks (SBRN'00)*, pages 173–178, Rio de Janeiro, RJ, Brazil, November 22–25, 2000. Washington, DC, USA: IEEE Computer Society. doi: 10.1109/SBRN.2000.889734.
22. Edward P. K. Tsang, Jin Li, Sheri Marina Markose, Hakan Er, Abdellah Salhi, and Guilia Iori. Eddie in financial decision making. *Journal of Management and Economics*, 4(4), November 2000. URL http://www.bracil.net/finance/papers/Tsang-Eddie-JMgtEcon2000.pdf.
23. Edward P. K. Tsang, Paul Yung, and Jin Li. Eddie-automation – a decision support tool for financial forecasting. *Decision Support Systems*, 37(4):559–565, September 2004. doi: 10.1016/S0167-9236(03)00087-3. URL http://www.bracil.net/finance/papers/TsYuLi-Eddie-Dss2004.pdf.
24. Pu Wang, Edward P. K. Tsang, Thomas Weise, Ke Tang, and Xin Yao. Using gp to evolve decision rules for classification in financial data sets. In Fuchun Sun, Yingxu Wang, Jianhua Lu, Bo Zhang, Witold Kinsner, and Lotfi A. Zadeh, editors, *Proceedings of the 9th IEEE International Conference on Cognitive Informatics (ICCI'10)*, pages 722–727, Beijing, China: Tsinghua University, July 7–9, 2010. Los Alamitos, CA, USA: IEEE Computer Society Press. doi: 10.1109/COGINF.2010.5599820.
25. Pu Wang, Thomas Weise, and Raymond Chiong. Novel evolutionary algorithms for supervised classification problems: An experimental study. *Evolutionary Intelligence*, 4(1):3–16, January 12, 2011. doi: 10.1007/s12065-010-0047-7.

Bibliography IV

26. Eleonora Bilotta, Antonio Cerasa, Pietro Pantano, Aldo Quattrone, Andrea Staino, and Francesca Stramandinoli. Evolving cellular neural networks for the automated segmentation of multiple sclerosis lesions. In Raymond Chiong, Thomas Weise, and Zbigniew Michalewicz, editors, *Variants of Evolutionary Algorithms for Real-World Applications*, pages 377–412. Berlin/Heidelberg: Springer-Verlag, 2011. doi: 10.1007/978-3-642-23424-8_12.
27. Faten Kharbat, Larry Bull, and Mohammed Odeh. Mining breast cancer data with xcs. In Dirk Thierens, Hans-Georg Beyer, Josh C. Bongard, Jürgen Branke, John Andrew Clark, Dave Cliff, Clare Bates Congdon, Kalyanmoy Deb, Benjamin Doerr, Tim Kovacs, Sanjeev P. Kumar, Julian Francis Miller, Jason H. Moore, Frank Neumann, Martin Pelikan, Riccardo Poli, Kumara Sastry, Kenneth Owen Stanley, Thomas Stützle, Richard A. Watson, and Ingo Wegener, editors, *Proceedings of 9th Genetic and Evolutionary Computation Conference (GECCO'07)*, pages 2066–2073, London, UK: University College London (UCL), July 7–11, 2007. New York, NY, USA: ACM Press. doi: 10.1145/1276958.1277362. URL `http://www.cs.york.ac.uk/rts/docs/GECCO_2007/docs/p2066.pdf`.
28. Shigeru Obayashi. Multidisciplinary design optimization of aircraft wing planform based on evolutionary algorithms. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC'98)*, volume 4, pages 3148–3153, La Jolla, CA, USA, October 11–14, 1998. Los Alamitos, CA, USA: IEEE Computer Society Press. doi: 10.1109/ICSMC.1998.726486. URL `http://www.lania.mx/~ccoello/obayashi98a.pdf.gz`.
29. Akira Oyama. *Wing Design Using Evolutionary Algorithm*. PhD thesis, Tokyo, Japan: Tokyo University, Department of Aeronautics and Space Engineering, March 2000. URL `http://flab.eng.isas.ac.jp/member/oyama/index2e.html`.
30. Miroslav Červenka and Vojtěch Křesálek. Aerodynamic wing optimisation using soma evolutionary algorithm. In Natalio Krasnogor, María Belén Melián-Batista, José Andrés Moreno Pérez, J. Marcos Moreno-Vega, and David Alejandro Pelta, editors, *Proceedings of the 3rd International Workshop Nature Inspired Cooperative Strategies for Optimization (NICSO'08)*, volume 236/2009 of *Studies in Computational Intelligence*, pages 127–138, Puerto de la Cruz, Tenerife, Spain, November 12–14, 2008. Berlin/Heidelberg: Springer-Verlag. doi: 10.1007/978-3-642-03211-0_11.
31. Miroslav Červenka and Ivan Zelinka. Application of evolutionary algorithm on aerodynamic wing optimisation. In *Proceedings of the 2nd European Computing Conference (ECC'08)*, Malta, September 11–13, 2008. URL `http://www.wseas.us/e-library/conferences/2008/malta/ecc/ecc53.pdf`.
32. Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, September 2003. doi: 10.1145/937503.937505. URL `http://iridia.ulb.ac.be/~meta/newsite/downloads/ACSUR-blum-roli.pdf`.

33. 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 0077092392, 0-470-22079-1, 0-632-03238-3, 9780077092399, 978-0470220795, and 978-0-632-03238-9. URL http://books.google.de/books?id=G6NfQgAACAAJ. Outgrowth of a conference held at Bangor, North Wales in 1990.
34. Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing: An International Journal*, 8(2):239–287, June 2009. doi: 10.1007/s11047-008-9098-4.
35. Nicos Christofides, Aristide Mingozzi, Paolo Toth, and C. Sandi, editors. *Combinatorial Optimization*. A Wiley-Interscience Publiction. New York, NY, USA: John Wiley & Sons Ltd., May 30–June 11, 1977. ISBN 0471997498 and 9780471997498. Based on a series of lectures given at the Summer School in Combinatorial Optimization held in SOGESTA, Urbino, Italy from 30th May to 11th June 1977.
36. William John Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization*. Estimation, Simulation, and Control – Wiley-Interscience Series in Discrete Mathematics and Optimization. Chichester, West Sussex, UK: Wiley Interscience, November 12, 1997. ISBN 0-471-55894-X and 978-0-471-55894-1.
37. Panos M. Pardalos and Ding-Zhu Du, editors. *Handbook of Combinatorial Optimization*, volume 1–3. Norwell, MA, USA: Kluwer Academic Publishers, October 31, 1990. ISBN 0-7923-5018-9, 0-7923-5019-7, 0-7923-5285-8, 0-7923-5293-9, 978-0-7923-5018-7, 978-0-7923-5019-4, 978-0-7923-5285-3, and 978-0-7923-5293-8.
38. Alexander Schrijver. *A Course in Combinatorial Optimization*. self-published, January 22, 2009. URL http://homepages.cwi.nl/~lex/files/dict.pdf.
39. Build your own 2013 760li sedan, September 13, 2012. URL http://www.bmwusa.com/Standard/Content/BYO/Byohome.aspx?NAModelCode=137K.
40. Xin Yao and Yong Liu. Scaling up evolutionary programming algorithms. In Vincent William Porto, N. Saravanan, D. Waagen, and Ágoston E. Eiben, editors, *Evolutionary Programming VII – Proceedings of the 7th International Conference on Evolutionary Programming (EP'98)*, volume 1447/1998 of *Lecture Notes in Computer Science (LNCS)*, pages 103–112, San Diego, CA, USA: Mission Valley Marriott, May 25–27, 1998. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/BFb0040764. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.3235.
41. Frank Hoffmeister and Thomas Bäck. Genetic algorithms and evolution strategies – similarities and differences. Technical Report SYS-1/92, Dortmund, North Rhine-Westphalia, Germany: Universität Dortmund, Fachbereich Informatik, Systems Analysis, 1992.

Bibliography VI

42. Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*, volume 18 of *Springer Series in Operations Research and Financial Engineering*. Berlin, Germany: Springer-Verlag GmbH, 2nd edition, 2006. ISBN 0-387-98793-2, 978-0-387-30303-1, and 978-0-387-98793-4. doi: 10.1007/978-0-387-40065-5. URL http://books.google.de/books?id=2B9LUYAP3n4C.
43. Juraj Hromkovič. *Algorithmics for Hard Computing Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics*. Texts in Theoretical Computer Science – An EATCS Series. Berlin, Germany: Springer-Verlag GmbH, 2001. ISBN 3-540-66860-8 and 978-3-540-44134-2. URL http://books.google.de/books?id=Ut1QAAAAMAAJ.
44. Wikipedia – the free encyclopedia, 2009. URL http://en.wikipedia.org/.
45. Alice1211. Millie's cookies recipe, August 13, 2011. URL http://www.bbcgoodfood.com/recipes/1580654/millies-cookies-recipe.
46. Fred W. Glover and Gary A. Kochenberger, editors. *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*. Norwell, MA, USA: Kluwer Academic Publishers, Dordrecht, Netherlands: Springer Netherlands, and Boston, MA, USA: Springer US, 2003. ISBN 0-306-48056-5, 1-4020-7263-5, 978-0-306-48056-0, and 978-1-4020-7263-5. doi: 10.1007/b101874. URL http://read.pudn.com/downloads140/sourcecode/others/606225/%EF%BC%81%EF%BC%81Springer%20-%20Handbook%20of%20Metaheuristics.pdf. Series Editor Frederick S. Hillier.
47. Zbigniew Michalewicz and David B. Fogel. *How to Solve It: Modern Heuristics*. Berlin/Heidelberg: Springer-Verlag, 2nd edition, 2004. ISBN 3-540-22494-7, 978-3-540-22494-5, and 978-3-642-06134-9. URL http://books.google.de/books?id=RJbV_-JlIUQC.
48. 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.
49. 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.
50. 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.

Bibliography VII

51. 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.
52. 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.
53. 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.
54. Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge International Series on Parallel Computation. Cambridge, UK: Cambridge University Press, 1995. ISBN 0-521-47465-5 and 978-0-521-47465-8. URL http://books.google.de/books?id=QKVY4mDivBEC.
55. Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge, UK: Cambridge University Press, 2005. ISBN 0-521-83540-2 and 978-0-521-83540-4. URL http://books.google.de/books?id=0bAYl6d7hvkC.
56. Juraj Hromkovič and I. Zámecniková. *Design and Analysis of Randomized Algorithms: Introduction to Design Paradigms*. Texts in Theoretical Computer Science – An EATCS Series. Berlin, Germany: Springer-Verlag GmbH, 2005. ISBN 3-540-23949-9, 978-3-540-23949-9, and 978-3-540-27903-7. doi: 10.1007/3-540-27903-2. URL http://books.google.de/books?id=EfuMfn8T5_oC.
57. Rajeev Motwani and Prabhakar Raghavan. Randomized algorithms. *ACM Computing Surveys (CSUR)*, 28(1):33–37, March 1996. doi: 10.1145/234313.234327.
58. Andrea G. B. Tettamanzi and Marco Tomassini. *Soft Computing: Integrating Evolutionary, Neural, and Fuzzy Systems*. Berlin, Germany: Springer-Verlag GmbH, 2001. ISBN 3-540-42204-8 and 978-3-540-42204-4. URL http://books.google.de/books?id=86jQw6v30KoC.

59. Frank Hoffmann, Mario Köppen, Frank Klawonn, and Rajkumar Roy, editors. *Soft Computing: Methodologies and Applications*, volume 32 of *Advances in Intelligent and Soft Computing*. Basel, Switzerland: Birkhäuser Verlag, May 2005. ISBN 3-540-25726-8 and 978-3-540-25726-4. doi: 10.1007/3-540-32400-3. URL http://books.google.de/books?id=ATswE4Z1m14C. Post-conference proceedings of the 8th Online World Conferences on Soft Computing (WSC8) which took place from September 29th to October 10th, 2003, organized by World Federation of Soft Computing (WFSC).
60. Lakhmi C. Jain and Janusz Kacprzyk, editors. *New Learning Paradigms in Soft Computing*, volume 84 (PR-200 of *Studies in Fuzziness and Soft Computing*. Berlin, Germany: Springer-Verlag GmbH, February 2002. ISBN 3-7908-1436-9 and 978-3-7908-1436-1. URL http://books.google.de/books?id=u2NcykOhYYwC.
61. Garg Deepak, editor. *Soft Computing*. Vadodara, Gujrat, India: Allied Publishers, 2005. ISBN 817764632X and 978-8177646320. URL http://books.google.de/books?id=IkajJC9iGxMC.
62. Carlos Artemio Coello Coello. A short tutorial on evolutionary multiobjective optimization. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos Artemio Coello Coello, and David Wolfe Corne, editors, *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO'01)*, volume 1993/2001 of *Lecture Notes in Computer Science (LNCS)*, pages 21–40, Zürich, Switzerland: Eidgenössische Technische Hochschule (ETH) Zürich, March 7–9, 2001. Berlin, Germany: Springer-Verlag GmbH. URL http://www.cs.cinvestav.mx/~emooworkgroup/tutorial-slides-coello.pdf.
63. Carlos Artemio Coello Coello. An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends. 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 1, pages 3–13, Washington, DC, USA: Mayflower Hotel, July 6–9, 1999. Piscataway, NJ, USA: IEEE Computer Society. doi: 10.1109/CEC.1999.781901. URL http://www-course.cs.york.ac.uk/evo/SupportingDocs/coellocoello99updated.pdf.
64. Carlos M. Fonseca and Peter J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, Spring 1995. doi: 10.1162/evco.1995.3.1.1. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.7779.
65. Carlos Artemio Coello Coello and Gary B. Lamont, editors. *Applications of Multi-Objective Evolutionary Algorithms*, volume 1 of *Advances in Natural Computation*. Singapore: World Scientific Publishing Co., December 2004. ISBN 981-256-106-4 and 978-981-256-106-0. URL http://books.google.de/books?id=viWm0k9meOcC.

Bibliography IX

66. Sanaz Mostaghim. *Multi-objective Evolutionary Algorithms: Data structures, Convergence and, Diversity*. PhD thesis, Paderborn, Germany: Universität Paderborn, Fakultät für Elektrotechnik, Informatik und Mathematik, February 2005. URL http://books.google.de/books?id=HoVLAAAACAAJ.
67. Carlos M. Fonseca and Peter J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms – part i: A unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 28(1):26–37, January 1998. doi: 10.1109/3468.650319. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.2473.
68. Carlos M. Fonseca and Peter J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms – part ii: Application example. Technical Report 565, Sheffield, UK: University of Sheffield, Department of Automatic Control and Systems Engineering, January 23, 1995. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.5395.
69. Carlos M. Fonseca and Peter J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms – part ii: Application example. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 28(1):38–47, January 1998. doi: 10.1109/3468.650320.
70. David H. Wolpert and William G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fé, NM, USA: Santa Fe Institute, February 6, 1995. URL http://www.santafe.edu/research/publications/workingpapers/95-02-010.pdf.
71. David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, 1(1):67–82, April 1997. doi: 10.1109/4235.585893. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.39.6926.
72. Anne Auger and Oliver Teytaud. Continuous lunches are free plus the design of optimal optimization algorithms. Rapports de Recherche inria-00369788, Institut National de Recherche en Informatique et en Automatique (INRIA), March 21, 2009. URL http://hal.inria.fr/docs/00/36/97/88/PDF/ccflRevisedVersionAugerTeytaud.pdf.
73. Anne Auger and Oliver Teytaud. Continuous lunches are free plus the design of optimal optimization algorithms. *Algorithmica*, 57(1):121–146, May 2010. doi: 10.1007/s00453-008-9244-5.