合肥学院
HEFEI UNIVERSITY

Optimization Algorithms

3. Metaheuristics

Thomas Weise · 汤卫思
tweise@hfuu.edu.cn · http://iao.hfuu.edu.cn/5

Institute of Applied Optimization (IAO) | 应用优化研究所
School of Artificial Intelligence and Big Data | 人工智能与大数据学院
Hefei University | 合肥学院
Hefei, Anhui, China | 中国安徽省合肥市

Outline

1. Introduction
2. Black-Box Characteristic
3. Summary

Introduction

Optimization Algorithms

- OK, by now we know already something about optimization problems and how we can give them a "structure."

Optimization Algorithms

- OK, by now we know already something about optimization problems and how we can give them a "structure."
- But how are they solved?

Optimization Algorithms

- OK, by now we know already something about optimization problems and how we can give them a "structure."
- But how are they solved?
- Optimization problems are solved by optimization algorithms.

Optimization Algorithms

- OK, by now we know already something about optimization problems and how we can give them a "structure."
- But how are they solved?
- Optimization problems are solved by optimization algorithms.
- Optimization algorithms can be divided into exact and heuristic methods.

Exact Algorithms

- Exact algorithms guarantee to find the optimal solution.

Exact Algorithms

- Exact algorithms guarantee to find the optimal solution if sufficient runtime is granted.

Exact Algorithms

- Exact algorithms guarantee to find the optimal solution if sufficient runtime is granted.
- This required runtime might, in the worst case, exceed what we can afford, in particular for $\mathcal{NP}$-hard problems, such as the JSSP.

Exact Algorithms

- Exact algorithms guarantee to find the optimal solution if sufficient runtime is granted.
- This required runtime might, in the worst case, exceed what we can afford, in particular for $\mathcal{NP}$-hard problems, such as the JSSP.
- Many exact methods can be halted before completing their run and they can then still provide an approximate solution (without the guarantee that it is optimal).

Heuristic Algorithms

- The idea behind heuristic algorithms is to get approximate solution relatively quickly.

Heuristic Algorithms

- The idea behind heuristic algorithms is to get approximate solution relatively quickly.
- They do not make any guarantees at all how good it will be.

Heuristic Algorithms

- The idea behind heuristic algorithms is to get approximate solution relatively quickly.
- They either do not make any guarantees at all how good it will be or, sometimes, provide some bound guarantee.

Heuristic Algorithms

- The idea behind heuristic algorithms is to get approximate solution relatively quickly.
- They either do not make any guarantees at all how good it will be or, sometimes, provide some bound guarantee (like: "This solution will not cost more than two times of the optimal cost.")

Heuristic Algorithms

- The idea behind heuristic algorithms is to get approximate solution relatively quickly.
- They either do not make any guarantees at all how good it will be or, sometimes, provide some bound guarantee (like: "This solution will not cost more than two times of the optimal cost.")
- Simple heuristics are usually tailor-made for specific problems, like the TSP or JSSP.

Metaheuristics

- Metaheuristics[1–5] are the center of this course.

Metaheuristics

- Metaheuristics[1–5] are the center of this course.

Definition (Metaheuristic)

A metaheuristic is a general algorithm that can produce approximate solutions for a class of different optimization problems.

Metaheuristics

- Metaheuristics[1–5] are the center of this course.

Definition (Metaheuristic)

A metaheuristic is a general algorithm that can produce approximate solutions for a class of different optimization problems.

- . . . and class is here considered in the wider sense and could even mean "all problems that can be presented in the structure we discussed in Lesson 2."

Metaheuristics

- Metaheuristics[1–5] are the center of this course.

Definition (Metaheuristic)

A metaheuristic is a general algorithm that can produce approximate solutions for a class of different optimization problems.

- . . . and class is here considered in the wider sense and could even mean "all problems that can be presented in the structure we discussed in Lesson 2."
- Because of this generality, they can be adapted to new optimization problems.

Metaheuristics

- Metaheuristics[1–5] are the center of this course.

Definition (Metaheuristic)

A metaheuristic is a general algorithm that can produce approximate solutions for a class of different optimization problems.

- ... and class is here considered in the wider sense and could even mean "all problems that can be presented in the structure we discussed in Lesson 2."
- Because of this generality, they can be adapted to new optimization problems.
- As long as we can bring the problem into the "structure" discussed before, we can attack it with a metaheuristic.

Metaheuristics

- Metaheuristics[1–5] are the center of this course.

Definition (Metaheuristic)

A metaheuristic is a general algorithm that can produce approximate solutions for a class of different optimization problems.

- ... and class is here considered in the wider sense and could even mean “all problems that can be presented in the structure we discussed in Lesson 2.”
- Because of this generality, they can be adapted to new optimization problems.
- As long as we can bring the problem into the “structure” discussed before, we can attack it with a metaheuristic.
- We will introduce several such general algorithms.

Metaheuristics

- Metaheuristics[1–5] are the center of this course.

Definition (Metaheuristic)

A metaheuristic is a general algorithm that can produce approximate solutions for a class of different optimization problems.

- . . . and class is here considered in the wider sense and could even mean "all problems that can be presented in the structure we discussed in Lesson 2."
- Because of this generality, they can be adapted to new optimization problems.
- As long as we can bring the problem into the "structure" discussed before, we can attack it with a metaheuristic.
- We will introduce several such general algorithms.
- We explore them by using the Job Shop Scheduling Problem (JSSP)[6–10] as example.

Black-Box Characteristic

Black-Box Optimization

- Why are metaheuristics general methods?

Black-Box Optimization

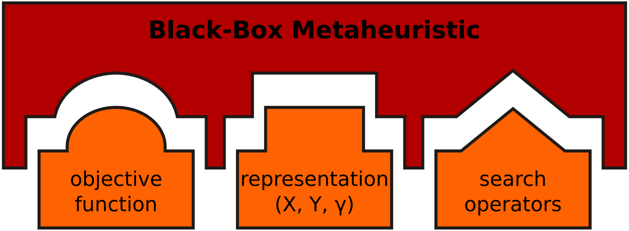
- Why are metaheuristics general methods?
- Because they allow us to divide between algorithm and problem.

Black-Box Optimization

- Why are metaheuristics general methods?
- Because they allow us to divide between algorithm and problem.
- From the algorithm perspective, optimization problems can be viewed as black boxes.

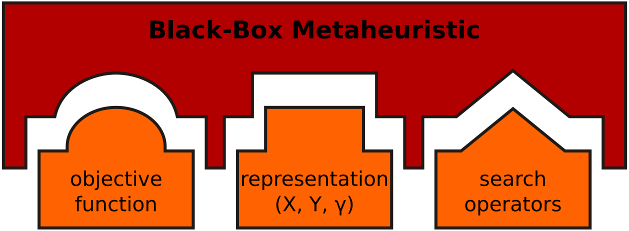
Black-Box Optimization

- Why are metaheuristics general methods?
- Because they allow us to divide between algorithm and problem.
- From the algorithm perspective, optimization problems can be viewed as black boxes.



Black-Box Optimization

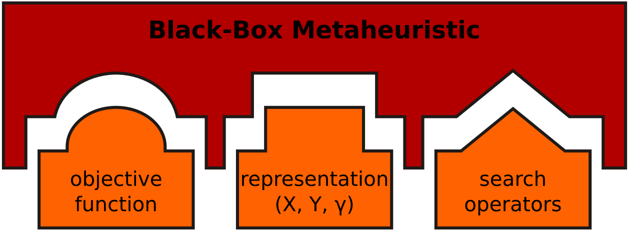
- Why are metaheuristics general methods?
- Because they allow us to divide between algorithm and problem.
- From the algorithm perspective, optimization problems can be viewed as black boxes.



- The metaheuristic does not care (much) how the objective function $f : \mathbb{Y} \mapsto \mathbb{R}$ is shaped.

Black-Box Optimization

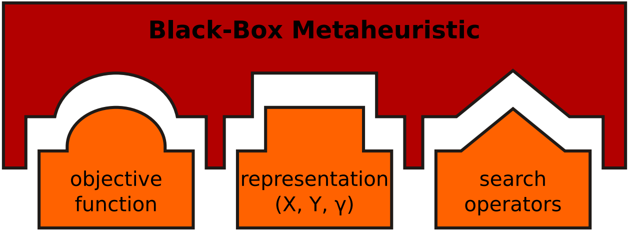
- Why are metaheuristics general methods?
- Because they allow us to divide between algorithm and problem.
- From the algorithm perspective, optimization problems can be viewed as black boxes.



- The metaheuristic does not care (much) how the objective function $f : \mathbb{Y} \mapsto \mathbb{R}$ is shaped.
- It does not care much about the structure of $\mathbb{Y}$, $\mathbb{X}$, and γ either.

Black-Box Optimization

- Why are metaheuristics general methods?
- Because they allow us to divide between algorithm and problem.
- From the algorithm perspective, optimization problems can be viewed as black boxes.



- The metaheuristic does not care (much) how the objective function $f : \mathbb{Y} \mapsto \mathbb{R}$ is shaped.
- It does not care much about the structure of $\mathbb{Y}$, $\mathbb{X}$, and γ either.
- We "plug them in" together with the search operators (about which we will talk later), and the metaheuristic will "work."

The Meaning of the Black-Box Characteristic

- This black-box character makes implementing metaheuristics complicated.

The Meaning of the Black-Box Characteristic

- This black-box character makes implementing metaheuristics complicated.
- If we look at "classical" algorithms in computer science, then they usually have clearly defined input and output data.

The Meaning of the Black-Box Characteristic

- This black-box character makes implementing metaheuristics complicated.
- If we look at "classical" algorithms in computer science, then they usually have clearly defined input and output data. Dijkstra's algorithm for shortest paths[11], for instance, accepts a graph with weighted edges and source node and returns the shortest paths to all other nodes.

The Meaning of the Black-Box Characteristic

- This black-box character makes implementing metaheuristics complicated.
- If we look at "classical" algorithms in computer science, then they usually have clearly defined input and output data. Dijkstra's algorithm for shortest paths[11], for instance, accepts a graph with weighted edges and source node and returns the shortest paths to all other nodes.
- This is also true for machine learning.

The Meaning of the Black-Box Characteristic

- This black-box character makes implementing metaheuristics complicated.
- If we look at "classical" algorithms in computer science, then they usually have clearly defined input and output data. Dijkstra's algorithm for shortest paths[11], for instance, accepts a graph with weighted edges and source node and returns the shortest paths to all other nodes.
- This is also true for machine learning. k-means clustering[12 13] expects a set of real vectors as input and returns k real vectors as output.

The Meaning of the Black-Box Characteristic

- This black-box character makes implementing metaheuristics complicated.
- If we look at "classical" algorithms in computer science, then they usually have clearly defined input and output data. Dijkstra's algorithm for shortest paths[11], for instance, accepts a graph with weighted edges and source node and returns the shortest paths to all other nodes.
- This is also true for machine learning. k-means clustering[12, 13] expects a set of real vectors as input and returns k real vectors as output. Deep learning[14] expects a labeled training set of real vectors and returns the weights of a neural network.

The Meaning of the Black-Box Characteristic

- This black-box character makes implementing metaheuristics complicated.
- If we look at "classical" algorithms in computer science, then they usually have clearly defined input and output data. Dijkstra's algorithm for shortest paths[11], for instance, accepts a graph with weighted edges and source node and returns the shortest paths to all other nodes.
- This is also true for machine learning. k-means clustering[12, 13] expects a set of real vectors as input and returns k real vectors as output. Deep learning[14] expects a labeled training set of real vectors and returns the weights of a neural network.
- Metaheuristics are nothing like that.

The Meaning of the Black-Box Characteristic

- This black-box character makes implementing metaheuristics complicated.
- If we look at "classical" algorithms in computer science, then they usually have clearly defined input and output data. Dijkstra's algorithm for shortest paths[11], for instance, accepts a graph with weighted edges and source node and returns the shortest paths to all other nodes.
- This is also true for machine learning. k-means clustering[12,13] expects a set of real vectors as input and returns k real vectors as output. Deep learning[14] expects a labeled training set of real vectors and returns the weights of a neural network.
- Metaheuristics are nothing like that.
- Matter of fact: Metaheuristics could be used for any of the above tasks!

The Meaning of the Black-Box Characteristic

- This black-box character makes implementing metaheuristics complicated.
- If we look at "classical" algorithms in computer science, then they usually have clearly defined input and output data. Dijkstra's algorithm for shortest paths[11], for instance, accepts a graph with weighted edges and source node and returns the shortest paths to all other nodes.
- This is also true for machine learning. k-means clustering[12 13] expects a set of real vectors as input and returns k real vectors as output. Deep learning[14] expects a labeled training set of real vectors and returns the weights of a neural network.
- Metaheuristics are nothing like that.
- Matter of fact: Metaheuristics could be used for any of the above tasks!
- The metaheuristics are general algorithms into which a representation fitting any of these tasks can be "plugged."

Implementing the Black Box Idea

- Catering for implementing algorithms that are so general seems to be quite hard, especially this early in the course.

Implementing the Black Box Idea

- Catering for implementing algorithms that are so general seems to be quite hard, especially this early in the course.
- But we already have the foundation: All the interfaces we discussed before!

Implementing the Black Box Idea

- Catering for implementing algorithms that are so general seems to be quite hard, especially this early in the course.
- But we already have the foundation: All the interfaces we discussed before!
- We just need to implement them and hand them to our algorithm implementations.

Implementing the Black Box Idea

- Catering for implementing algorithms that are so general seems to be quite hard, especially this early in the course.
- But we already have the foundation: All the interfaces we discussed before!
- We just need to implement them and hand them to our algorithm implementations.
- For this, I provide one abstraction: the interface `IBlackBoxProcess`.

Implementing the Black Box Idea

- Catering for implementing algorithms that are so general seems to be quite hard, especially this early in the course.
- But we already have the foundation: All the interfaces we discussed before!
- We just need to implement them and hand them to our algorithm implementations.
- For this, I provide one abstraction: the interface `IBlackBoxProcess`.
- I will not discuss here how exactly it is implemented, but we will take a quick peek on what it can do.

`IBlackBoxProcess`...

- provides a random number generator to the algorithm

`IBlackBoxProcess`...

- provides a random number generator to the algorithm,
- wraps an objective function f together with a representation mapping γ to allow us to evaluate a point in the search space $x \in \mathbb{X}$ in a single step, effectively performing $f(\gamma(x))$

`IBlackBoxProcess`...

- provides a random number generator to the algorithm,
- wraps an objective function f together with a representation mapping γ to allow us to evaluate a point in the search space $x \in \mathbb{X}$ in a single step, effectively performing $f(\gamma(x))$,
- keeps track of the elapsed runtime and FEs as well as when the last improvement was made by updating said information when necessary during the invocations of the "wrapped" objective function

`IBlackBoxProcess`...

- provides a random number generator to the algorithm,
- wraps an objective function f together with a representation mapping γ to allow us to evaluate a point in the search space $x \in \mathbb{X}$ in a single step, effectively performing $f(\gamma(x))$,
- keeps track of the elapsed runtime and FEs as well as when the last improvement was made by updating said information when necessary during the invocations of the "wrapped" objective function,
- keeps track of the best points in the search space and solution space so far as well as their associated objective value in special variables by updating them whenever the "wrapped" objective function discovers an improvement

`IBlackBoxProcess`...

- provides a random number generator to the algorithm,
- wraps an objective function f together with a representation mapping γ to allow us to evaluate a point in the search space $x \in \mathbb{X}$ in a single step, effectively performing $f(\gamma(x))$,
- keeps track of the elapsed runtime and FEs as well as when the last improvement was made by updating said information when necessary during the invocations of the "wrapped" objective function,
- keeps track of the best points in the search space and solution space so far as well as their associated objective value in special variables by updating them whenever the "wrapped" objective function discovers an improvement,
- represents a termination criterion (e.g., maximum FEs, maximum runtime, reaching a goal objective value)

`IBlackBoxProcess`...

- provides a random number generator to the algorithm,
- wraps an objective function f together with a representation mapping γ to allow us to evaluate a point in the search space $x \in \mathbb{X}$ in a single step, effectively performing $f(\gamma(x))$,
- keeps track of the elapsed runtime and FEs as well as when the last improvement was made by updating said information when necessary during the invocations of the "wrapped" objective function,
- keeps track of the best points in the search space and solution space so far as well as their associated objective value in special variables by updating them whenever the "wrapped" objective function discovers an improvement,
- represents a termination criterion (e.g., maximum FEs, maximum runtime, reaching a goal objective value), and
- logs the improvements that the algorithm makes to a text file, so that we can use them to make tables and draw diagrams.

IBlackBoxProcess

```
package aitoa.structure;

public interface IBlackBoxProcess<X, Y> extends
    IObjectiveFunction<X>, // evaluate works on x ∈ 𝕏 and performs γ
    ITerminationCriterion, // shouldTerminate() tells when to stop
    Closeable { // when closed, can write log file with trace

  Random getRandom(); // replicable random numbers
// ...
  double getBestF();      // get (current best or end) quality
  void getBestX(X dest); // get (current best or end) x ∈ 𝕏
  void getBestY(Y dest); // get (current best or end) y ∈ 𝕐
// ...
  long getConsumedFEs();        // get number of calls to evaluate
  long getLastImprovementFE(); // get last FE when improved

/** Some stuff that is not relevant here has been omitted.
    You can find it in the full code online. */
}
```

Summary

Summary

- Now we finally have all the components together to implement metaheuristic optimization algorithms!

谢谢

Thank you

References I

1. Thomas Weise. *An Introduction to Optimization Algorithms*. Institute of Applied Optimization (IAO) [应用优化研究所] of the School of Artificial Intelligence and Big Data [人工智能与大数据学院] of Hefei University [合肥学院], Hefei [合肥市], Anhui [安徽省], China [中国], 2018–2020. URL http://thomasweise.github.io/aitoa/.
2. Thomas Weise. *Global Optimization Algorithms – Theory and Application*. it-weise.de (self-published), Germany, 2009. URL http://www.it-weise.de/projects/book.pdf.
3. Michel Gendreau and Jean-Yves Potvin, editors. *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science (ISOR)*. Springer Science+Business Media, LLC, Boston, MA, USA, 2 edition, 2010. ISBN 978-1-4419-1663-1. doi:10.1007/978-1-4419-1665-5.
4. Fred Glover and Gary A. Kochenberger, editors. *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science (ISOR)*. Springer Netherlands, Dordrecht, Netherlands, 2003. ISBN 0-306-48056-5. doi:10.1007/b101874.
5. Bastien Chopard and Marco Tomassini. *An Introduction to Metaheuristics for Optimization*, volume 54 of *Natural Computing Series (NCS)*. Springer Nature Switzerland AG, Cham, Switzerland, 2018. doi:10.1007/978-3-319-93073-2.
6. Ronald Lewis Graham, Eugene Leighton Lawler, Jan Karel Lenstra, and Alexander Hendrik George Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5: 287–326, 1979. doi:10.1016/S0167-5060(08)70356-X.
7. Eugene Leighton Lawler, Jan Karel Lenstra, Alexander Hendrik George Rinnooy Kan, and David B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In Stephen C. Graves, Alexander Hendrik George Rinnooy Kan, and Paul H. Zipkin, editors, *Handbook of Operations Research and Management Science*, volume IV: Production Planning and Inventory, chapter 9, pages 445–522. North-Holland Scientific Publishers Ltd., Amsterdam, The Netherlands, 1993. doi:10.1016/S0927-0507(05)80189-6.
8. Eugene Leighton Lawler. Recent results in the theory of machine scheduling. In AAchim Bachem, Bernhard Korte, and Martin Grötschel, editors, *Math Programming: The State of the Art*, chapter 8, pages 202–234. Springer-Verlag, Bonn/New York, 1982. ISBN 978-3-642-68876-8. doi:10.1007/978-3-642-68874-4_9.
9. Éric D. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research (EJOR)*, 64(2): 278–285, January 1993. doi:10.1016/0377-2217(93)90182-M.
10. Jacek Błażewicz, Wolfgang Domschke, and Erwin Pesch. The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research (EJOR)*, 93:1–33, August 1996. doi:10.1016/0377-2217(95)00362-2. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.159.1650&type=pdf.
11. Edsger Wybe Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. doi:10.1007/BF01386390. URL http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.165.7577&rep=rep1&type=pdf.

References II

12. Edward W. Forgy. Cluster analysis of multivariate data: Efficiency vs interpretability of classifications. *Biometrics*, 21(3): 768–780, 1965.
13. John A. Hartigan and Manchek A. Wong. Algorithm AS 136: A K-Means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979. doi:10.2307/2346830.
14. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. URL http://www.deeplearningbook.org.