COMP 5660/6660 - Evolutionary Computing - Lecture Slides

Daniel Tauritz, PhD

Auburn University

August 20, 2025

- Step 1: build abstract/computational model of the real-world

[1] https://quoteinvestigator.com/2011/05/13/einstein-simple/

Computational Problem Solving

- Step 1: build abstract/computational model of the real-world
- Step 2: solve computationally in abstract model

Computational Problem Solving

- Step 1: build abstract/computational model of the real-world
- Step 2: solve computationally in abstract model
- "Everything Should Be Made as Simple as Possible, But Not Simpler"[1]
- Step 3: map solution back to real-world

[1] `https://quoteinvestigator.com/2011/05/13/einstein-simple/`

Computational Problem Classes

- Decision problems (e.g. SAT)

Computational Problem Classes

- Decision problems (e.g. SAT)
- Optimization problems (e.g., TSP, MAXSAT)

Computational Problem Classes

- Decision problems (e.g. SAT)
- Optimization problems (e.g., TSP, MAXSAT)
- Modeling (aka system identification) problems (e.g., ML)

Computational Problem Classes

- Decision problems (e.g. SAT)
- Optimization problems (e.g., TSP, MAXSAT)
- Modeling (aka system identification) problems (e.g., ML)
- Simulation problems

Computational Problem Classes

- Decision problems (e.g. SAT)
- Optimization problems (e.g., TSP, MAXSAT)
- Modeling (aka system identification) problems (e.g., ML)
- Simulation problems
- Search problems

Computational Problem Classes

- Decision problems (e.g. SAT)
- Optimization problems (e.g., TSP, MAXSAT)
- Modeling (aka system identification) problems (e.g., ML)
- Simulation problems
- Search problems
- Mad Engineer Thought Experiment

Search Terminology

- Many computational problems can be formulated as **generate-and-test** search problems

Search Terminology

- Many computational problems can be formulated as **generate-and-test** search problems
- A **search space** contains the set of all possible solutions

Search Terminology

- Many computational problems can be formulated as **generate-and-test** search problems
- A **search space** contains the set of all possible solutions
- A **search space generator** is *complete* if it can generate the entire search space

Search Terminology

- Many computational problems can be formulated as **generate-and-test** search problems
- A **search space** contains the set of all possible solutions
- A **search space generator** is *complete* if it can generate the entire search space
- An **objective function** tests the quality of a solution

Search Terminology

- Many computational problems can be formulated as **generate-and-test** search problems
- A **search space** contains the set of all possible solutions
- A **search space generator** is *complete* if it can generate the entire search space
- An **objective function** tests the quality of a solution
- Graduated solution quality

Search Terminology

- Many computational problems can be formulated as **generate-and-test** search problems
- A **search space** contains the set of all possible solutions
- A **search space generator** is *complete* if it can generate the entire search space
- An **objective function** tests the quality of a solution
- Graduated solution quality
- Stochastic search of adaptive solution landscape

Search Terminology

- Many computational problems can be formulated as **generate-and-test** search problems
- A **search space** contains the set of all possible solutions
- A **search space generator** is *complete* if it can generate the entire search space
- An **objective function** tests the quality of a solution
- Graduated solution quality
- Stochastic search of adaptive solution landscape
- Local versus global optima

Search Terminology

- Many computational problems can be formulated as **generate-and-test** search problems
- A **search space** contains the set of all possible solutions
- A **search space generator** is *complete* if it can generate the entire search space
- An **objective function** tests the quality of a solution
- Graduated solution quality
- Stochastic search of adaptive solution landscape
- Local versus global optima
- Unimodal versus multimodal problems

Algorithmic Toolbox

- A **heuristic** is a problem-dependent rule-of-thumb

Algorithmic Toolbox

- A **heuristic** is a problem-dependent rule-of-thumb
- A **meta-heuristic** is a general heuristic to determine the sampling order over a search space with the goal to find a near-optimal solution (or set of solutions)

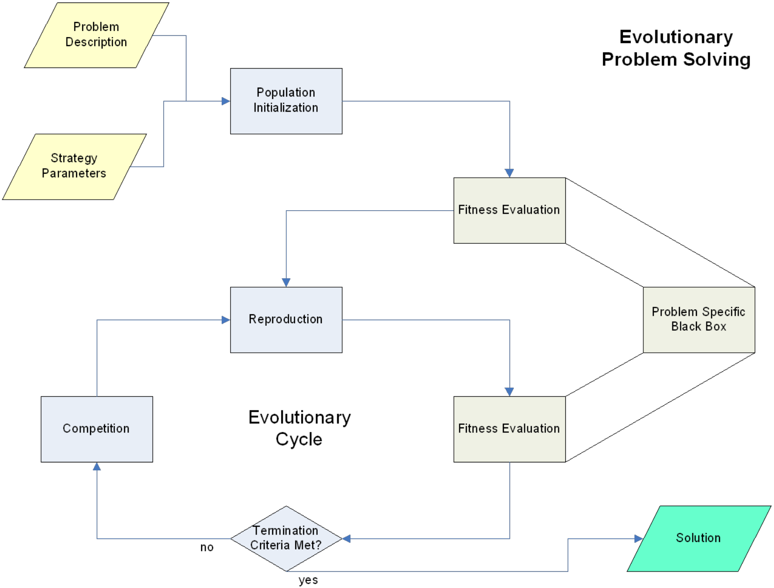
Algorithmic Toolbox

- A **heuristic** is a problem-dependent rule-of-thumb
- A **meta-heuristic** is a general heuristic to determine the sampling order over a search space with the goal to find a near-optimal solution (or set of solutions)
- A **hyper-heuristic** is a meta-heuristic for a space of programs
- A **Black-Box Search Algorithm (BBSA)** is a meta-heuristic which iteratively generates trial solutions employing solely the information gained from previous trial solutions, but no explicit problem knowledge

Algorithmic Toolbox

- A **heuristic** is a problem-dependent rule-of-thumb
- A **meta-heuristic** is a general heuristic to determine the sampling order over a search space with the goal to find a near-optimal solution (or set of solutions)
- A **hyper-heuristic** is a meta-heuristic for a space of programs
- A **Black-Box Search Algorithm (BBSA)** is a meta-heuristic which iteratively generates trial solutions employing solely the information gained from previous trial solutions, but no explicit problem knowledge
- **Evolutionary Algorithms (EAs)** can be described as a class of *stochastic*, *population-based* BBSAs inspired by *Evolution Theory*, *Genetics*, and *Population Dynamics*

Evolutionary Cycle



EA Pros

- More general purpose than traditional optimization algorithms (less problem specific knowledge required)

EA Pros

- More general purpose than traditional optimization algorithms (less problem specific knowledge required)
- Ability to solve “difficult” problems

EA Pros

- More general purpose than traditional optimization algorithms (less problem specific knowledge required)
- Ability to solve "difficult" problems
- Solution availability during computation
- Robustness
- Inherent parallelism

EA Cons

- Fitness function and genetic operators often not obvious
- Premature convergence
- Computationally intensive
- Difficult parameter optimization

Biological Metaphors - Darwinian Evolution

- Macroscopic view of evolution
- Natural selection
- Survival of the fittest
- Random variation
- Genetic drift

Biological Metaphors - Mendelian Genetics

- Genotype - functional unit of inheritance

Biological Metaphors - Mendelian Genetics

- Genotype - functional unit of inheritance
- Phenotype - expression of genotype

Biological Metaphors - Mendelian Genetics

- Genotype - functional unit of inheritance
- Phenotype - expression of genotype
- Pleiotropy - one gene affects multiple phenotypic traits

Biological Metaphors - Mendelian Genetics

- Genotype - functional unit of inheritance
- Phenotype - expression of genotype
- Pleiotropy - one gene affects multiple phenotypic traits
- Polygeny - one phenotypic trait is affected by multiple genes

Biological Metaphors - Mendelian Genetics

- Genotype - functional unit of inheritance
- Phenotype - expression of genotype
- Pleiotropy - one gene affects multiple phenotypic traits
- Polygeny - one phenotypic trait is affected by multiple genes
- Chromosomes - haploid versus diploid

Biological Metaphors - Mendelian Genetics

- Genotype - functional unit of inheritance
- Phenotype - expression of genotype
- Pleiotropy - one gene affects multiple phenotypic traits
- Polygeny - one phenotypic trait is affected by multiple genes
- Chromosomes - haploid versus diploid
- Locus/Loci - gene location/locations on the genome/chromosome

Biological Metaphors - Mendelian Genetics

- Genotype - functional unit of inheritance
- Phenotype - expression of genotype
- Pleiotropy - one gene affects multiple phenotypic traits
- Polygeny - one phenotypic trait is affected by multiple genes
- Chromosomes - haploid versus diploid
- Locus/Loci - gene location/locations on the genome/chromosome
- Alleles - variant forms of a gene

Nature versus digital realm

- Environment - Problem search space

Nature versus digital realm

- Environment - Problem search space
- Environmental fitness - Fitness Function

Nature versus digital realm

- Environment - Problem search space
- Environmental fitness - Fitness Function
- Population - Set

Nature versus digital realm

- Environment - Problem search space
- Environmental fitness - Fitness Function
- Population - Set
- Individual - Datastructure

Nature versus digital realm

- Environment - Problem search space
- Environmental fitness - Fitness Function
- Population - Set
- Individual - Datastructure
- Genes - Elements

Nature versus digital realm

- Environment - Problem search space
- Environmental fitness - Fitness Function
- Population - Set
- Individual - Datastructure
- Genes - Elements
- Alleles - Datatype

Assignment Series 1 Genotype-Phenotype Mapping

- The genotype is a fixed-length linear representation, with len(*shapes*) genes.

Assignment Series 1 Genotype-Phenotype Mapping

- The genotype is a fixed-length linear representation, with len(*shapes*) genes.
- Each gene is a 3-tuple (x, y, r) specifying the location & rotation of a specific shape.

Assignment Series 1 Genotype-Phenotype Mapping

- The genotype is a fixed-length linear representation, with len(*shapes*) genes.
- Each gene is a 3-tuple (x, y, r) specifying the location & rotation of a specific shape.
- Hence an allele is a tuple of three values, with x and y within the bounds of the stock, and r within the range 0 to 3.

Assignment Series 1 Genotype-Phenotype Mapping

- The genotype is a fixed-length linear representation, with len(*shapes*) genes.
- Each gene is a 3-tuple (x, y, r) specifying the location & rotation of a specific shape.
- Hence an allele is a tuple of three values, with x and y within the bounds of the stock, and r within the range 0 to 3.
- So a single integer value of x, y, or r is NOT a full allele in our interpretation.

Assignment Series 1 Genotype-Phenotype Mapping

- The genotype is a fixed-length linear representation, with len(*shapes*) genes.
- Each gene is a 3-tuple (x, y, r) specifying the location & rotation of a specific shape.
- Hence an allele is a tuple of three values, with x and y within the bounds of the stock, and r within the range 0 to 3.
- So a single integer value of x, y, or r is NOT a full allele in our interpretation.
- Some possible phenotypes are:
  1. a matrix indicating for each cell which shapes overlap it

Assignment Series 1 Genotype-Phenotype Mapping

- The genotype is a fixed-length linear representation, with len(*shapes*) genes.
- Each gene is a 3-tuple (x, y, r) specifying the location & rotation of a specific shape.
- Hence an allele is a tuple of three values, with x and y within the bounds of the stock, and r within the range 0 to 3.
- So a single integer value of x, y, or r is NOT a full allele in our interpretation.
- Some possible phenotypes are:
  1. a matrix indicating for each cell which shapes overlap it
  2. a matrix indicating for each cell how many shapes overlap it

Assignment Series 1 Genotype-Phenotype Mapping

- The genotype is a fixed-length linear representation, with len(*shapes*) genes.
- Each gene is a 3-tuple (x, y, r) specifying the location & rotation of a specific shape.
- Hence an allele is a tuple of three values, with x and y within the bounds of the stock, and r within the range 0 to 3.
- So a single integer value of x, y, or r is NOT a full allele in our interpretation.
- Some possible phenotypes are:
  1. a matrix indicating for each cell which shapes overlap it
  2. a matrix indicating for each cell how many shapes overlap it
  3. a matrix indicating for each cell whether it's not overlapped, overlapped by one shape, or overlapped by multiple shapes

Genospace versus phenospace

- Genotype space

Genospace versus phenospace

- Genotype space
- Phenoype space

Genospace versus phenospace

- Genotype space
- Phenoype space
- Encoding & Decoding

Genospace versus phenospace

- Genotype space
- Phenoype space
- Encoding & Decoding
- Eight/N-Queens Problem

Genospace versus phenospace

- Genotype space
- Phenoype space
- Encoding & Decoding
- Eight/N-Queens Problem
- Choice of representation

Genospace-phenospace mapping

Genospace-phenospace mapping

Let F be the decoder function from G (genospace) to P (phenospace) and x^* be the global optimum.

Genospace-phenospace mapping

Let F be the decoder function from G (genospace) to P (phenospace) and x^* be the global optimum.

- $F : G \rightarrow P$ is *surjective* if $\forall p \in P \exists g \in G : F(g) = p$

Genospace-phenospace mapping

Let F be the decoder function from G (genospace) to P (phenospace) and x^* be the global optimum.

- $F : G \rightarrow P$ is *surjective* if $\forall p \in P \exists g \in G : F(g) = p$
- $F : G \rightarrow P$ is *injective* if $\forall g_1, g_2 \in G (F(g_1) = F(g_2)) \Rightarrow (g_1 = g_2)$

Genospace-phenospace mapping

Let F be the decoder function from G (genospace) to P (phenospace) and x^* be the global optimum.

- $F : G \to P$ is *surjective* if $\forall p \in P \exists g \in G : F(g) = p$
- $F : G \to P$ is *injective* if $\forall g_1, g_2 \in G (F(g_1) = F(g_2)) \Rightarrow (g_1 = g_2)$
- $F : G \to P$ is *bijective* if F is surjective and injective

Genospace-phenospace mapping

Let F be the decoder function from G (genospace) to P (phenospace) and x^* be the global optimum.

- $F : G \rightarrow P$ is *surjective* if $\forall p \in P \exists g \in G : F(g) = p$
- $F : G \rightarrow P$ is *injective* if $\forall g_1, g_2 \in G(F(g_1) = F(g_2)) \Rightarrow (g_1 = g_2)$
- $F : G \rightarrow P$ is *bijective* if F is surjective and injective
- If F is not surjective and $x^* \notin F(G)$, then the EA cannot find the global optimum. Therefore one should think twice before choosing a non-surjective decoder function if one cannot guarantee that the global optimum is still reachable.

Genospace-phenospace mapping

Let F be the decoder function from G (genospace) to P (phenospace) and x^* be the global optimum.

- $F : G \rightarrow P$ is *surjective* if $\forall p \in P \exists g \in G : F(g) = p$
- $F : G \rightarrow P$ is *injective* if $\forall g_1, g_2 \in G(F(g_1) = F(g_2)) \Rightarrow (g_1 = g_2)$
- $F : G \rightarrow P$ is *bijective* if F is surjective and injective
- If F is not surjective and $x^* \notin F(G)$, then the EA cannot find the global optimum. Therefore one should think twice before choosing a non-surjective decoder function if one cannot guarantee that the global optimum is still reachable.
- F does not need to be injective, but realize there is less to search if F is injective so there should be sufficient compensation, such as limiting F(G) to valid solutions in a constraint satisfaction problem.

The 0-1 Knapsack Problem

Given a set of n items with values v_i and cost c_i, select a subset that maximises value while not exceeding the capacity limit C_{max}.

The 0-1 Knapsack Problem

Given a set of n items with values v_i and cost c_i, select a subset that maximises value while not exceeding the capacity limit C_{max}.We consider two cases:

1. $fitness(p) = \sum_{i=1}^{n}(v_i \cdot g_i)$

The 0-1 Knapsack Problem

Given a set of n items with values v_i and cost c_i, select a subset that maximises value while not exceeding the capacity limit C_{max}.We consider two cases:

1. $fitness(p) = \sum_{i=1}^{n}(v_i \cdot g_i)$
2. Modify $fitness(p)$ to exclude items that would exceed C_{max}

Problem solving steps

- Collect problem knowledge

Problem solving steps

- Collect problem knowledge
- Choose gene representation

Problem solving steps

- Collect problem knowledge
- Choose gene representation
- Design fitness function

Problem solving steps

- Collect problem knowledge
- Choose gene representation
- Design fitness function
- Creation of initial population

Problem solving steps

- Collect problem knowledge
- Choose gene representation
- Design fitness function
- Creation of initial population
- Parent selection

Problem solving steps

- Collect problem knowledge
- Choose gene representation
- Design fitness function
- Creation of initial population
- Parent selection
- Decide on genetic operators

Problem solving steps

- Collect problem knowledge
- Choose gene representation
- Design fitness function
- Creation of initial population
- Parent selection
- Decide on genetic operators
- Competition / survival

Problem solving steps

- Collect problem knowledge
- Choose gene representation
- Design fitness function
- Creation of initial population
- Parent selection
- Decide on genetic operators
- Competition / survival
- Choose termination condition

Problem solving steps

- Collect problem knowledge
- Choose gene representation
- Design fitness function
- Creation of initial population
- Parent selection
- Decide on genetic operators
- Competition / survival
- Choose termination condition
- Find good parameter values

Permutation Representations

- Order based (e.g., job shop scheduling)

Permutation Representations

- Order based (e.g., job shop scheduling)
- Adjacency based (e.g., TSP)

Permutation Representations

- Order based (e.g., job shop scheduling)
- Adjacency based (e.g., TSP)
- Encodings
  - Event space: [A,B,C,D]

Permutation Representations

- Order based (e.g., job shop scheduling)
- Adjacency based (e.g., TSP)
- Encodings
  - Event space: [A,B,C,D]
  - Permutation: [3,1,2,4]

Permutation Representations

- Order based (e.g., job shop scheduling)
- Adjacency based (e.g., TSP)
- Encodings
  - Event space: [A,B,C,D]
  - Permutation: [3,1,2,4]
  - Mapping 1: [C,A,B,D]

Permutation Representations

- Order based (e.g., job shop scheduling)
- Adjacency based (e.g., TSP)
- Encodings
  - Event space: [A,B,C,D]
  - Permutation: [3,1,2,4]
  - Mapping 1: [C,A,B,D] $\rightarrow$ allele in locus i indicates event in that place in the sequence

Permutation Representations

- Order based (e.g., job shop scheduling)
- Adjacency based (e.g., TSP)
- Encodings
  - Event space: [A,B,C,D]
  - Permutation: [3,1,2,4]
  - Mapping 1: [C,A,B,D] $\rightarrow$ allele in locus i indicates event in that place in the sequence
  - Mapping 2: [B,C,A,D]

Permutation Representations

- Order based (e.g., job shop scheduling)
- Adjacency based (e.g., TSP)
- Encodings
  - Event space: [A,B,C,D]
  - Permutation: [3,1,2,4]
  - Mapping 1: [C,A,B,D] $\rightarrow$ allele in locus i indicates event in that place in the sequence
  - Mapping 2: [B,C,A,D] $\rightarrow$ allele in locus i indicates when the i^{th} event takes place