Evolutionary Computing
Lecture 1
Introduction. Simple Genetic Algorithm

Danylo Tavrov

Outline

Outline

Instructor

- Danylo Tavrov
- Education:
  - 2011: Applied Mathematics Department, BSc
  - 2013: Applied Mathematics Department, MSc
  - 2016: Applied Mathematics Department, PhD
  - 2019: Kyiv School of Economics, MA
  - 2022: University of California, Berkeley, MA in Economics
- Teaching at the Applied Mathematics Department since 2013 (with some breaks)
- Contacts:

Instructor

- Danylo Tavrov
- Education:
  - 2011: Applied Mathematics Department, BSc
  - 2013: Applied Mathematics Department, MSc
  - 2016: Applied Mathematics Department, PhD
  - 2019: Kyiv School of Economics, MA
  - 2022: University of California, Berkeley, MA in Economics
- Teaching at the Applied Mathematics Department since 2013 (with some breaks)
- Contacts:

Instructor

- Danylo Tavrov
- Education:
  - 2011: Applied Mathematics Department, BSc
  - 2013: Applied Mathematics Department, MSc
  - 2016: Applied Mathematics Department, PhD
  - 2019: Kyiv School of Economics, MA
  - 2022: University of California, Berkeley, MA in Economics
- Teaching at the Applied Mathematics Department since 2013 (with some breaks)
- Contacts:

Instructor

- Danylo Tavrov
- Education:
  - 2011: Applied Mathematics Department, BSc
  - 2013: Applied Mathematics Department, MSc
  - 2016: Applied Mathematics Department, PhD
  - 2019: Kyiv School of Economics, MA
  - 2022: University of California, Berkeley, MA in Economics
- Teaching at the Applied Mathematics Department since 2013 (with some breaks)
- Contacts:

Instructor

- Danylo Tavrov
- Education:
  - 2011: Applied Mathematics Department, BSc
  - 2013: Applied Mathematics Department, MSc
  - 2016: Applied Mathematics Department, PhD
  - 2019: Kyiv School of Economics, MA
  - 2022: University of California, Berkeley, MA in Economics
- Teaching at the Applied Mathematics Department since 2013 (with some breaks)
- Contacts:

Instructor

- Danylo Tavrov
- Education:
  - 2011: Applied Mathematics Department, BSc
  - 2013: Applied Mathematics Department, MSc
  - 2016: Applied Mathematics Department, PhD
  - 2019: Kyiv School of Economics, MA
  - 2022: University of California, Berkeley, MA in Economics
- Teaching at the Applied Mathematics Department since 2013 (with some breaks)
- Contacts:

Instructor

- Danylo Tavrov
- Education:
  - 2011: Applied Mathematics Department, BSc
  - 2013: Applied Mathematics Department, MSc
  - 2016: Applied Mathematics Department, PhD
  - 2019: Kyiv School of Economics, MA
  - 2022: University of California, Berkeley, MA in Economics
- Teaching at the Applied Mathematics Department since 2013 (with some breaks)
- Contacts:
  - d.tavrov@kpi.ua
  - Office hours — by appointment

Instructor

- Danylo Tavrov
- Education:
  - 2011: Applied Mathematics Department, BSc
  - 2013: Applied Mathematics Department, MSc
  - 2016: Applied Mathematics Department, PhD
  - 2019: Kyiv School of Economics, MA
  - 2022: University of California, Berkeley, MA in Economics
- Teaching at the Applied Mathematics Department since 2013 (with some breaks)
- Contacts:
  - d.tavrov@kpi.ua
  - Office hours — by appointment

Instructor

- Danylo Tavrov
- Education:
  - 2011: Applied Mathematics Department, BSc
  - 2013: Applied Mathematics Department, MSc
  - 2016: Applied Mathematics Department, PhD
  - 2019: Kyiv School of Economics, MA
  - 2022: University of California, Berkeley, MA in Economics
- Teaching at the Applied Mathematics Department since 2013 (with some breaks)
- Contacts:
  - d.tavrov@kpi.ua
  - Office hours — by appointment

Instructor

- Danylo Tavrov
- Education:
  - 2011: Applied Mathematics Department, BSc
  - 2013: Applied Mathematics Department, MSc
  - 2016: Applied Mathematics Department, PhD
  - 2019: Kyiv School of Economics, MA
  - 2022: University of California, Berkeley, MA in Economics
- Teaching at the Applied Mathematics Department since 2013 (with some breaks)
- Contacts:
  - d.tavrov@kpi.ua
  - Office hours — by appointment

Instructor

- Danylo Tavrov
- Education:
  - 2011: Applied Mathematics Department, BSc
  - 2013: Applied Mathematics Department, MSc
  - 2016: Applied Mathematics Department, PhD
  - 2019: Kyiv School of Economics, MA
  - 2022: University of California, Berkeley, MA in Economics
- Teaching at the Applied Mathematics Department since 2013 (with some breaks)
- Contacts:
  - d.tavrov@kpi.ua
  - Office hours — by appointment

Course Materials

- All materials are uploaded to a Github repository:
  - Syllabus (although it is more formal rather than useful)
  - Lecture slides
  - Literature, videos, and other useful stuff
- All lectures and one to one meetings are held in Zoom

Course Materials

- All materials are uploaded to a Github repository:
  - Syllabus (although it is more formal rather than useful)
  - Lecture slides
  - Literature, videos, and other useful stuff
- All lectures and one to one meetings are held in Zoom

Course Materials

- All materials are uploaded to a Github repository:
  - Syllabus (although it is more formal rather than useful)
  - Lecture slides
  - Literature, videos, and other useful stuff
- All lectures and one to one meetings are held in Zoom

Course Materials

- All materials are uploaded to a Github repository:
  - Syllabus (although it is more formal rather than useful)
  - Lecture slides
  - Literature, videos, and other useful stuff
- All lectures and one to one meetings are held in Zoom

Course Materials

- All materials are uploaded to a Github repository:
  - Syllabus (although it is more formal rather than useful)
  - Lecture slides
  - Literature, videos, and other useful stuff
- All lectures and one to one meetings are held in Zoom

Course Structure

- We will have **three (four???)** lectures:
  - Introduction and basic facts about evolutionary algorithms
  - Genetic algorithms
  - Evolution strategies
  - Memetic algorithms (and some applications from personal experience)
- The idea of this small course is to help students analyze how evolutionary algorithms **can be used in their PhD theses**
- Therefore, instead of sections we will have **one to one** meetings, discussing specifics of everyone's work
- At the end of the class, each student will **present** his evolutionary algorithm
- Successful presentation will substitute a formal exam

Course Structure

- We will have **three (four???)** lectures:
  - Introduction and basic facts about evolutionary algorithms
  - Genetic algorithms
  - Evolution strategies
  - Memetic algorithms (and some applications from personal experience)
- The idea of this small course is to help students analyze how evolutionary algorithms **can be used in their PhD theses**
- Therefore, instead of sections we will have **one to one** meetings, discussing specifics of everyone's work
- At the end of the class, each student will **present** his evolutionary algorithm
- Successful presentation will substitute a formal exam

Course Structure

- We will have **three (four???)** lectures:
  - Introduction and basic facts about evolutionary algorithms
  - Genetic algorithms
  - Evolution strategies
  - Memetic algorithms (and some applications from personal experience)
- The idea of this small course is to help students analyze how evolutionary algorithms **can be used in their PhD theses**
- Therefore, instead of sections we will have **one to one** meetings, discussing specifics of everyone's work
- At the end of the class, each student will **present** his evolutionary algorithm
- Successful presentation will substitute a formal exam

Course Structure

- We will have **three (four???)** lectures:
  - Introduction and basic facts about evolutionary algorithms
  - Genetic algorithms
  - Evolution strategies
  - Memetic algorithms (and some applications from personal experience)
- The idea of this small course is to help students analyze how evolutionary algorithms **can be used in their PhD theses**
- Therefore, instead of sections we will have **one to one** meetings, discussing specifics of everyone's work
- At the end of the class, each student will **present** his evolutionary algorithm
- Successful presentation will substitute a formal exam

Course Structure

- We will have **three (four???)** lectures:
  - Introduction and basic facts about evolutionary algorithms
  - Genetic algorithms
  - Evolution strategies
  - Memetic algorithms (and some applications from personal experience)
- The idea of this small course is to help students analyze how evolutionary algorithms **can be used in their PhD theses**
- Therefore, instead of sections we will have **one to one** meetings, discussing specifics of everyone's work
- At the end of the class, each student will **present** his evolutionary algorithm
- Successful presentation will substitute a formal exam

Course Structure

- We will have **three (four???)** lectures:
  - Introduction and basic facts about evolutionary algorithms
  - Genetic algorithms
  - Evolution strategies
  - Memetic algorithms (and some applications from personal experience)
- The idea of this small course is to help students analyze how evolutionary algorithms **can be used in their PhD theses**
- Therefore, instead of sections we will have **one to one** meetings, discussing specifics of everyone's work
- At the end of the class, each student will **present** his evolutionary algorithm
- Successful presentation will substitute a formal exam

Course Structure

- We will have **three (four???)** lectures:
  - Introduction and basic facts about evolutionary algorithms
  - Genetic algorithms
  - Evolution strategies
  - Memetic algorithms (and some applications from personal experience)
- The idea of this small course is to help students analyze how evolutionary algorithms **can be used in their PhD theses**
- Therefore, instead of sections we will have **one to one** meetings, discussing specifics of everyone's work
- At the end of the class, each student will **present** his evolutionary algorithm
- Successful presentation will substitute a formal exam

Course Structure

- We will have **three (four???)** lectures:
  - Introduction and basic facts about evolutionary algorithms
  - Genetic algorithms
  - Evolution strategies
  - Memetic algorithms (and some applications from personal experience)
- The idea of this small course is to help students analyze how evolutionary algorithms **can be used in their PhD theses**
- Therefore, instead of sections we will have **one to one** meetings, discussing specifics of everyone's work
- At the end of the class, each student will **present** his evolutionary algorithm
- Successful presentation will substitute a formal exam

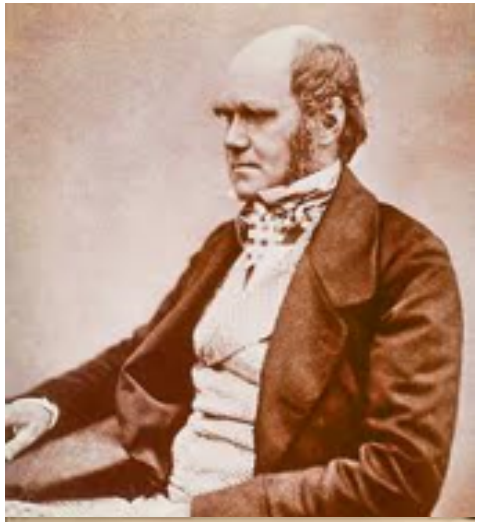
Course Structure

- We will have **three (four???)** lectures:
  - Introduction and basic facts about evolutionary algorithms
  - Genetic algorithms
  - Evolution strategies
  - Memetic algorithms (and some applications from personal experience)
- The idea of this small course is to help students analyze how evolutionary algorithms **can be used in their PhD theses**
- Therefore, instead of sections we will have **one to one** meetings, discussing specifics of everyone's work
- At the end of the class, each student will **present** his evolutionary algorithm
- Successful presentation will substitute a formal exam

Outline

Theory of Evolution

- Evolution is the most fundamental theory in biology
- Many people don't understand evolution
- Evolutionary Computing is not a part of biology
- It only takes inspiration from the process of Darwinian evolution
- *Watch video*



ON

THE ORIGIN OF SPECIES

BY MEANS OF NATURAL SELECTION,

OR THE

PRESERVATION OF FAVOURED RACES IN THE STRUGGLE FOR LIFE.

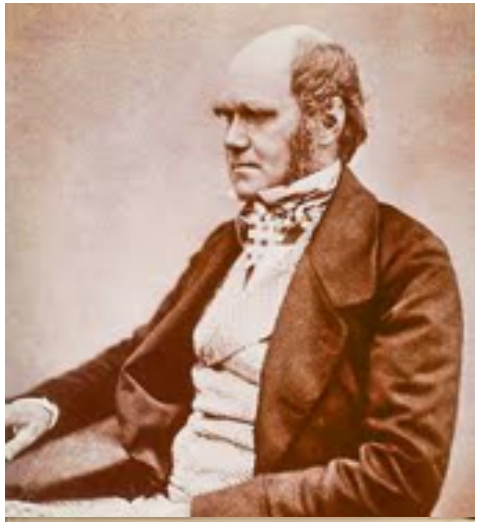
By CHARLES DARWIN, M.A.,

LONDON:
JOHN MURRAY, ALBEMARLE STREET.
1859.

- Charles Darwin (1809–1882) introduced the notion of **natural selection** in his book *On the Origin of Species by Means of Natural Selection* (Murray, London, 1859)
- He sought to explain how modern species can evolve from earlier species
- This was his great idea: not the idea of evolution, but the idea of evolution by natural selection

Theory of Evolution

- Evolution is the most fundamental theory in biology
- Many people don't understand evolution
- Evolutionary Computing is not a part of biology
- It only takes inspiration from the process of Darwinian evolution
- *Watch video*



ON

THE ORIGIN OF SPECIES

BY MEANS OF NATURAL SELECTION,

OR THE

PRESERVATION OF FAVOURED RACES IN THE STRUGGLE FOR LIFE.

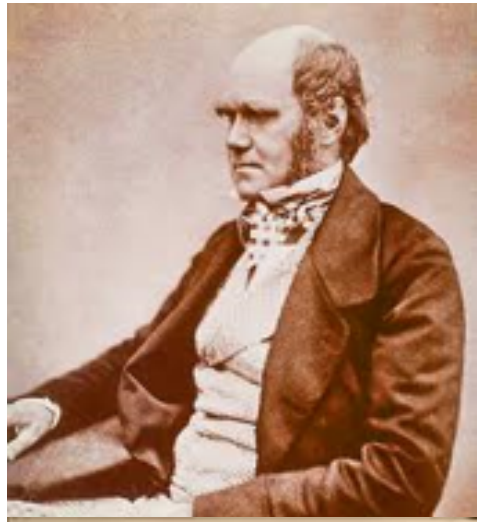
By CHARLES DARWIN, M.A.,

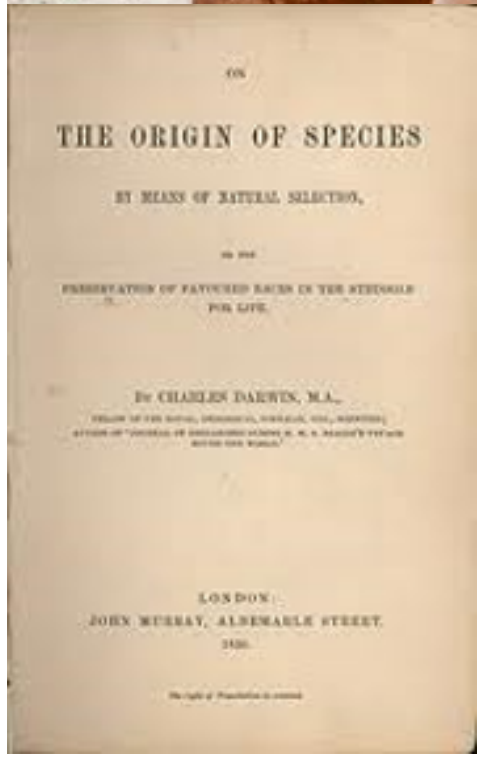
LONDON:
JOHN MURRAY, ALBEMARLE STREET.

- Charles Darwin (1809–1882) introduced the notion of **natural selection** in his book *On the Origin of Species by Means of Natural Selection* (Murray, London, 1859)
- He sought to explain how modern species can evolve from earlier species
- This was his great idea: not the idea of evolution, but the idea of evolution by natural selection

Theory of Evolution

- Evolution is the most fundamental theory in biology
- Many people don't understand evolution
- Evolutionary Computing is not a part of biology
- It only takes inspiration from the process of Darwinian evolution
- *Watch video*





ON

THE ORIGIN OF SPECIES

BY MEANS OF NATURAL SELECTION,

OR THE

PRESERVATION OF FAVOURED RACES IN THE STRUGGLE FOR LIFE.

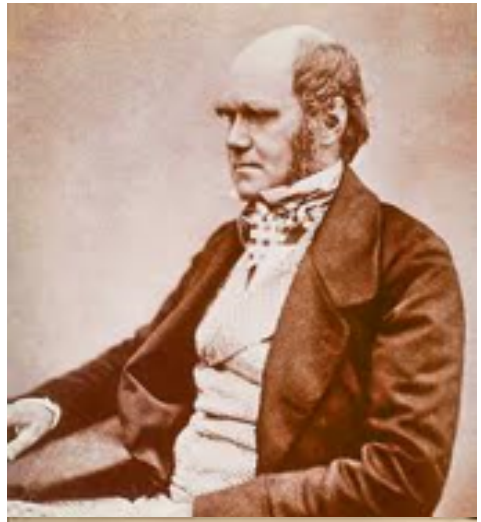
By CHARLES DARWIN, M.A.,

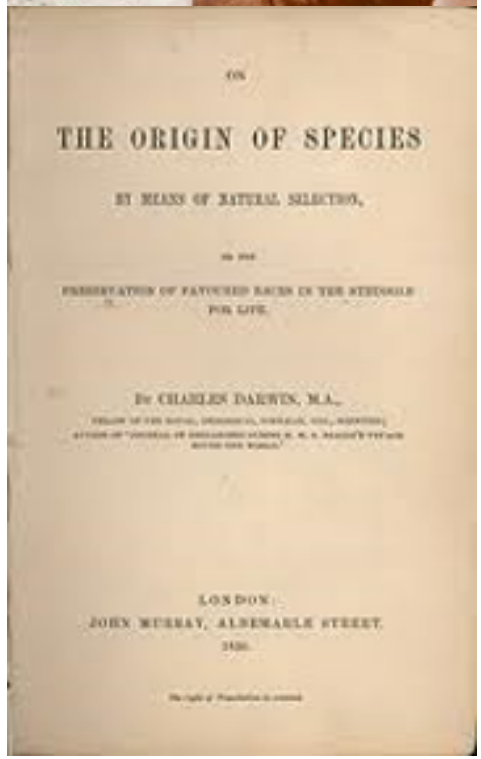
LONDON:
JOHN MURRAY, ALBEMARLE STREET.

- Charles Darwin (1809–1882) introduced the notion of **natural selection** in his book *On the Origin of Species by Means of Natural Selection* (Murray, London, 1859)
- He sought to explain how modern species can evolve from earlier species
- This was his great idea: not the idea of evolution, but the idea of evolution by natural selection

Theory of Evolution

- Evolution is the most fundamental theory in biology
- Many people don't understand evolution
- Evolutionary Computing is not a part of biology
- It only takes inspiration from the process of Darwinian evolution
- *Watch video*
- Charles Darwin (1809–1882) introduced the notion of **natural selection** in his book *On the Origin of Species by Means of Natural Selection* (Murray, London, 1859)
- He sought to explain how modern species can evolve from earlier species
- This was his great idea: not the idea of evolution, but the idea of evolution by natural selection




ON

THE ORIGIN OF SPECIES

BY MEANS OF NATURAL SELECTION,

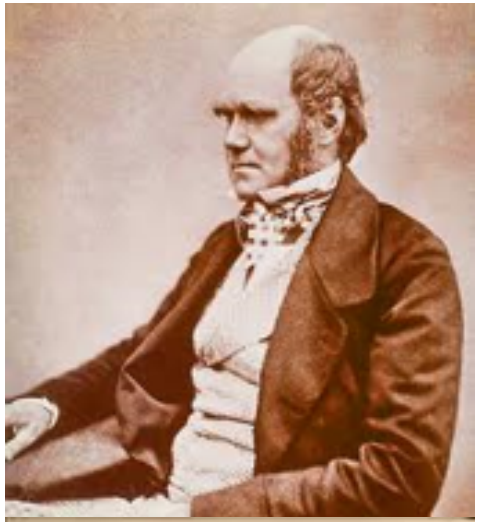
By CHARLES DARWIN, M.A.,

LONDON:
JOHN MURRAY, ALBEMARLE STREET.

Theory of Evolution

- Evolution is the most fundamental theory in biology
- Many people don't understand evolution
- Evolutionary Computing is not a part of biology
- It only takes inspiration from the process of Darwinian evolution
- *Watch video*

- Charles Darwin (1809–1882) introduced the notion of **natural selection** in his book *On the Origin of Species by Means of Natural Selection* (Murray, London, 1859)
- He sought to explain how modern species can evolve from earlier species
- This was his great idea: not the idea of evolution, but the idea of evolution by **natural selection**




ON

THE ORIGIN OF SPECIES

BY MEANS OF NATURAL SELECTION,

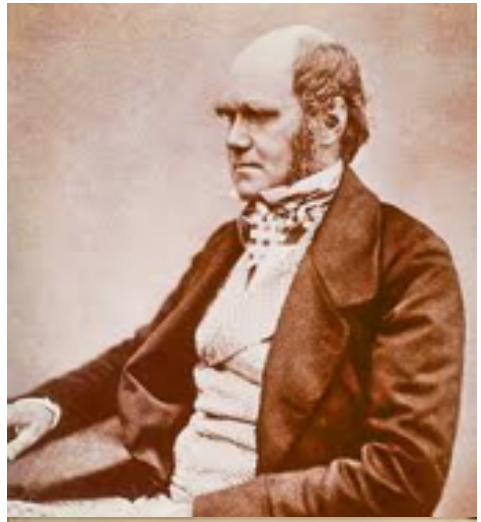
By CHARLES DARWIN, M.A.,

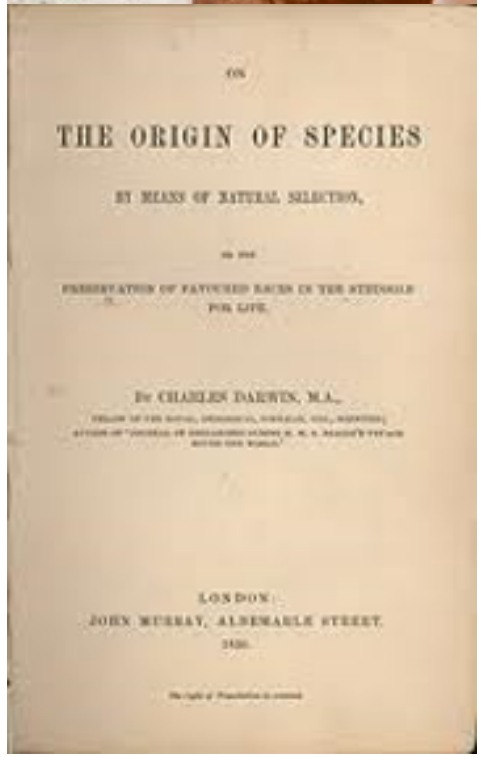
LONDON:
JOHN MURRAY, ALBEMARLE STREET.

Theory of Evolution

- Evolution is the most fundamental theory in biology
- Many people don't understand evolution
- Evolutionary Computing is not a part of biology
- It only takes inspiration from the process of Darwinian evolution
- *Watch video*

- Charles Darwin (1809–1882) introduced the notion of **natural selection** in his book *On the Origin of Species by Means of Natural Selection* (Murray, London, 1859)
- He sought to explain how modern species can evolve from earlier species
- This was his great idea: not the idea of evolution, but the idea of evolution by **natural selection**




ON

THE ORIGIN OF SPECIES

BY MEANS OF NATURAL SELECTION,

OR THE

PRESERVATION OF FAVOURED RACES IN THE STRUGGLE FOR LIFE.

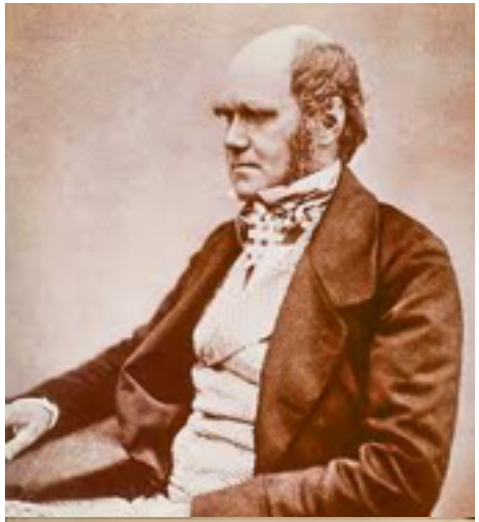
By CHARLES DARWIN, M.A.,

LONDON:
JOHN MURRAY, ALBEMARLE STREET.

Theory of Evolution

- Evolution is the most fundamental theory in biology
- Many people don't understand evolution
- Evolutionary Computing is not a part of biology
- It only takes inspiration from the process of Darwinian evolution
- *Watch video*

- Charles Darwin (1809–1882) introduced the notion of **natural selection** in his book *On the Origin of Species by Means of Natural Selection* (Murray, London, 1859)
- He sought to explain how modern species can evolve from earlier species
- This was his great idea: not the idea of evolution, but the idea of evolution by **natural selection**




ON

THE ORIGIN OF SPECIES

BY MEANS OF NATURAL SELECTION,

OR THE

PRESERVATION OF FAVOURED RACES IN THE STRUGGLE FOR LIFE.

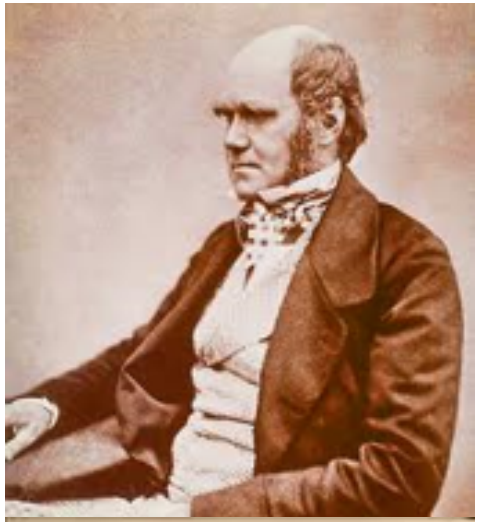
By CHARLES DARWIN, M.A.,

LONDON:
JOHN MURRAY, ALBEMARLE STREET.

Theory of Evolution

- Evolution is the most fundamental theory in biology
- Many people don't understand evolution
- Evolutionary Computing is not a part of biology
- It only takes inspiration from the process of Darwinian evolution
- *Watch video*

- Charles Darwin (1809–1882) introduced the notion of **natural selection** in his book *On the Origin of Species by Means of Natural Selection* (Murray, London, 1859)
- He sought to explain how modern species can evolve from earlier species
- This was his great idea: not the idea of evolution, but the idea of evolution by **natural selection**




ON

THE ORIGIN OF SPECIES

BY MEANS OF NATURAL SELECTION,

OR THE

PRESERVATION OF FAVOURED RACES IN THE STRUGGLE FOR LIFE.

By CHARLES DARWIN, M.A.,

LONDON:
JOHN MURRAY, ALBEMARLE STREET.

Natural Selection

- All environments have finite resources, therefore some kind of **selection** is inevitable
- Those individuals that compete for the resources most effectively increase chance of reproduction
  - "Survival of the fittest," after Herbert Spencer, 1820–1903)
- Phenotypic traits can lead to higher chances of reproduction and can be inherited
- A **population** is a diverse set of **individuals**
- **Combinations of traits** that are better adapted (**fit**) tend to increase representation in population
- **Variations** occur through **random changes** during reproduction yielding constant source of diversity
- **Selective pressure** favors one trait over another

Natural Selection

- All environments have finite resources, therefore some kind of **selection** is inevitable
- Those individuals that compete for the resources most effectively increase chance of reproduction
  - "Survival of the fittest," after Herbert Spencer, 1820–1903)
- Phenotypic traits can lead to higher chances of reproduction and can be inherited
  - Then they will tend to increase in subsequent generations
    Leading to new combinations of traits
- A **population** is a diverse set of **individuals**
- **Combinations of traits** that are better adapted (**fit**) tend to increase representation in population
- **Variations** occur through **random changes** during reproduction yielding constant source of diversity
- **Selective pressure** favors one trait over another

Natural Selection

- All environments have finite resources, therefore some kind of **selection** is inevitable
- Those individuals that compete for the resources most effectively increase chance of reproduction
  - "Survival of the fittest," after Herbert Spencer, 1820–1903)
- Phenotypic traits can lead to higher chances of reproduction and can be inherited
  - Then they will tend to increase in subsequent generations
  - Leading to new combinations of traits
- A **population** is a diverse set of **individuals**
- **Combinations of traits** that are better adapted (**fit**) tend to increase representation in population
- **Variations** occur through **random changes** during reproduction yielding constant source of diversity
- **Selective pressure** favors one trait over another

Natural Selection

- All environments have finite resources, therefore some kind of **selection** is inevitable
- Those individuals that compete for the resources most effectively increase chance of reproduction
  - "Survival of the fittest," after Herbert Spencer, 1820–1903)
- Phenotypic traits can lead to higher chances of reproduction and can be inherited
  - Then they will tend to increase in subsequent generations
  - Leading to new combinations of traits
- A **population** is a diverse set of **individuals**
- **Combinations of traits** that are better adapted (**fit**) tend to increase representation in population
- **Variations** occur through **random changes** during reproduction yielding constant source of diversity
- **Selective pressure** favors one trait over another

Natural Selection

- All environments have finite resources, therefore some kind of **selection** is inevitable
- Those individuals that compete for the resources most effectively increase chance of reproduction
  - "Survival of the fittest," after Herbert Spencer, 1820–1903)
- Phenotypic traits can lead to higher chances of reproduction and can be inherited
  - Then they will tend to increase in subsequent generations
  - Leading to new combinations of traits
- A **population** is a diverse set of **individuals**
- **Combinations of traits** that are better adapted (**fit**) tend to increase representation in population
- **Variations** occur through **random changes** during reproduction yielding constant source of diversity
- **Selective pressure** favors one trait over another

Natural Selection

- All environments have finite resources, therefore some kind of **selection** is inevitable
- Those individuals that compete for the resources most effectively increase chance of reproduction
  - "Survival of the fittest," after Herbert Spencer, 1820–1903)
- Phenotypic traits can lead to higher chances of reproduction and can be inherited
  - Then they will tend to increase in subsequent generations
  - Leading to new combinations of traits
- A **population** is a diverse set of **individuals**
- **Combinations of traits** that are better adapted (**fit**) tend to increase representation in population
- **Variations** occur through **random changes** during reproduction yielding constant source of diversity
- **Selective pressure** favors one trait over another

Natural Selection

- All environments have finite resources, therefore some kind of **selection** is inevitable
- Those individuals that compete for the resources most effectively increase chance of reproduction
  - "Survival of the fittest," after Herbert Spencer, 1820–1903)
- Phenotypic traits can lead to higher chances of reproduction and can be inherited
  - Then they will tend to increase in subsequent generations
  - Leading to new combinations of traits
- A **population** is a diverse set of **individuals**
- **Combinations of traits** that are better adapted (**fit**) tend to increase representation in population
- **Variations** occur through **random changes** during reproduction yielding constant source of diversity
- **Selective pressure** favors one trait over another

Natural Selection

- All environments have finite resources, therefore some kind of **selection** is inevitable
- Those individuals that compete for the resources most effectively increase chance of reproduction
  - "Survival of the fittest," after Herbert Spencer, 1820–1903)
- Phenotypic traits can lead to higher chances of reproduction and can be inherited
  - Then they will tend to increase in subsequent generations
  - Leading to new combinations of traits
- A **population** is a diverse set of **individuals**
- **Combinations of traits** that are better adapted (**fit**) tend to increase representation in population
- **Variations** occur through **random changes** during reproduction yielding constant source of diversity
- **Selective pressure** favors one trait over another

Natural Selection

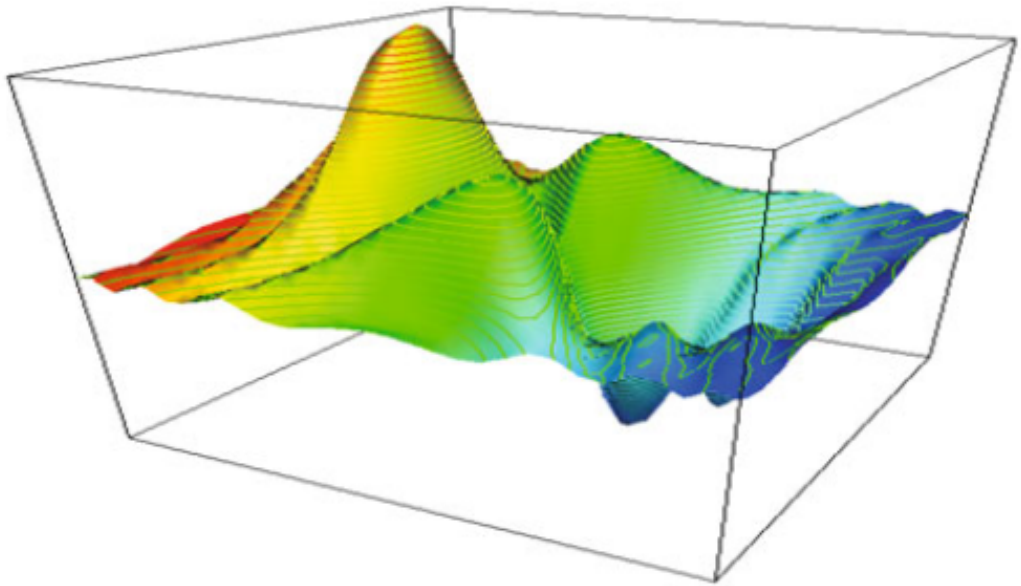
- All environments have finite resources, therefore some kind of **selection** is inevitable
- Those individuals that compete for the resources most effectively increase chance of reproduction
  - "Survival of the fittest," after Herbert Spencer, 1820–1903)
- Phenotypic traits can lead to higher chances of reproduction and can be inherited
  - Then they will tend to increase in subsequent generations
  - Leading to new combinations of traits
- A **population** is a diverse set of **individuals**
- **Combinations of traits** that are better adapted (**fit**) tend to increase representation in population
- **Variations** occur through **random changes** during reproduction yielding constant source of diversity
- **Selective pressure** favors one trait over another

Natural Selection

- All environments have finite resources, therefore some kind of **selection** is inevitable
- Those individuals that compete for the resources most effectively increase chance of reproduction
  - "Survival of the fittest," after Herbert Spencer, 1820–1903)
- Phenotypic traits can lead to higher chances of reproduction and can be inherited
  - Then they will tend to increase in subsequent generations
  - Leading to new combinations of traits
- A **population** is a diverse set of **individuals**
- **Combinations of traits** that are better adapted (**fit**) tend to increase representation in population
- **Variations** occur through **random changes** during reproduction yielding constant source of diversity
- **Selective pressure** favors one trait over another

Fitness Landscape

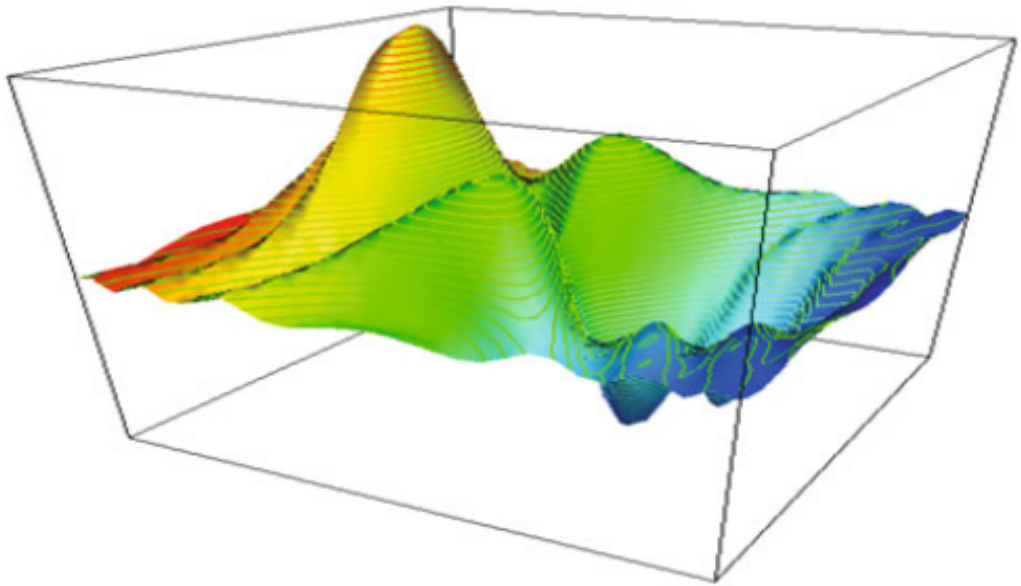
- We can imagine a population with n traits as existing in a $(n+1)$-dimensional space (landscape)
  - With height corresponding to fitness of an individual
- Selection pushes the population up the landscape
- Eventually, the fitness of the whole population increases
- *Watch video*



Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015), Fig. 2.2

Fitness Landscape

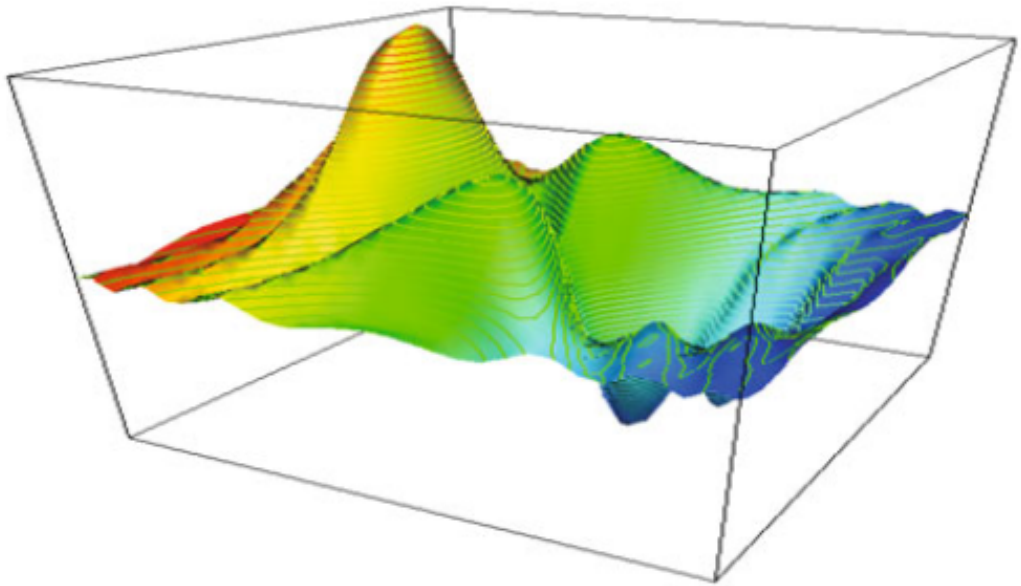
- We can imagine a population with n traits as existing in a $(n+1)$-dimensional space (landscape)
  - With height corresponding to fitness of an individual
- Selection pushes the population up the landscape
- Eventually, the fitness of the whole population increases
- *Watch video*



Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015), Fig. 2.2

Fitness Landscape

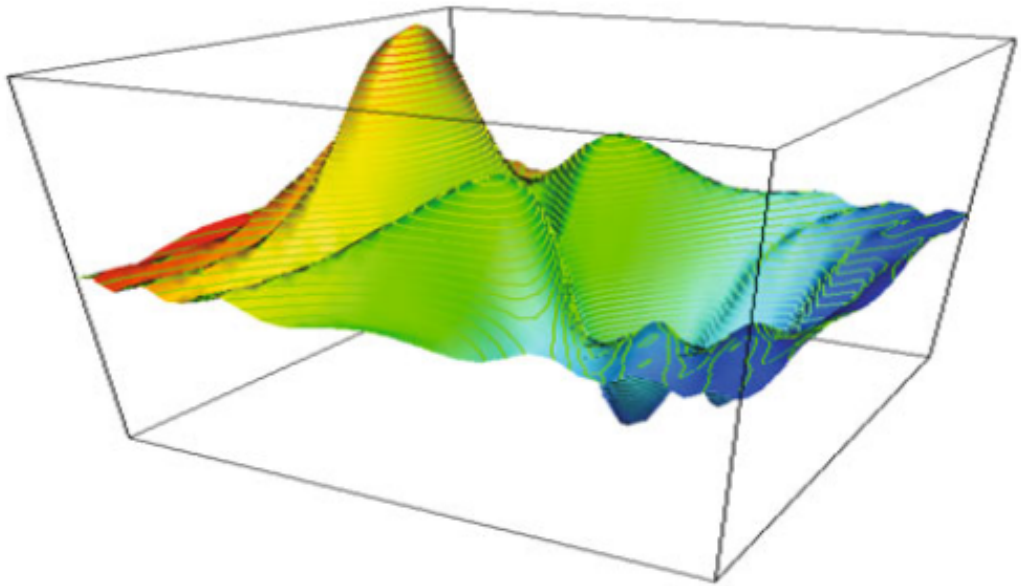
- We can imagine a population with n traits as existing in a $(n+1)$-dimensional space (landscape)
  - With height corresponding to fitness of an individual
- Selection pushes the population up the landscape
- Eventually, the fitness of the whole population increases
- *Watch video*



Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015), Fig. 2.2

Fitness Landscape

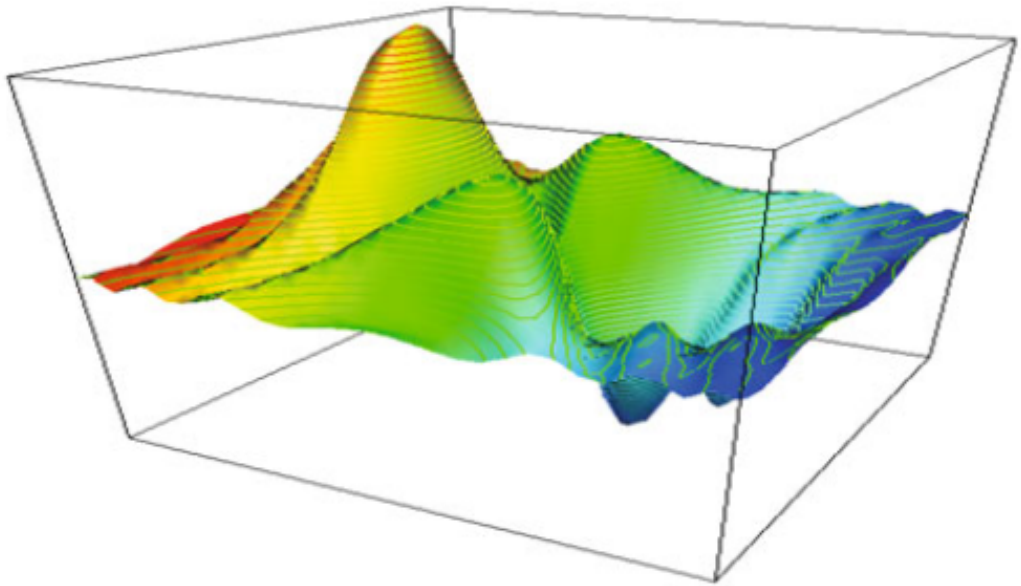
- We can imagine a population with n traits as existing in a $(n+1)$-dimensional space (landscape)
  - With height corresponding to fitness of an individual
- Selection pushes the population up the landscape
- Eventually, the fitness of the whole population increases
- *Watch video*



Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015), Fig. 2.2

Fitness Landscape

- We can imagine a population with n traits as existing in a $(n+1)$-dimensional space (landscape)
  - With height corresponding to fitness of an individual
- Selection pushes the population up the landscape
- Eventually, the fitness of the whole population increases
- *Watch video*



Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015), Fig. 2.2

Natural Selection and Genetics

- According to the geneticist Steve Jones[1],

  > *Had [Darwin] published his [book] today he would have been in trouble with the Trades Description Act because if there is one thing which The Origin of Species is not about, it is the origin of species.*

- Darwin didn't know what was the unit of natural selection, namely, **genes**
- The information required to build a living organism is coded in the DNA of that organism (four nucleotides, A, G, T, C, in a double helix spiral)
- Nucleotides encode specific amino acids, out of which all proteins are composed
- R. Dawkins's *River out of Eden*: "[...] genes [...] are long strings of pure digital information. [...] The machine code of genes is uncannily computerlike"
- Genes are encoded in strands of DNA called **chromosomes** (23 pairs in a human body)
- **Genotype** determines **phenotype** via a complex mapping (not one to one)

1. https://www.lrb.co.uk/the-paper/v15/n08/steve-jones/a-slower-kind-of-bang

Natural Selection and Genetics

- According to the geneticist Steve Jones[1],

  Had [Darwin] published his [book] today he would have been in trouble with the Trades Description Act because if there is one thing which The Origin of Species is not about, it is the origin of species.

- Darwin didn't know what was the unit of natural selection, namely, **genes**
- The information required to build a living organism is coded in the DNA of that organism (four nucleotides, A, G, T, C, in a double helix spiral)
- Nucleotides encode specific amino acids, out of which all proteins are composed
- R. Dawkins's *River out of Eden*: "[...] genes [...] are long strings of pure digital information. [...] The machine code of genes is uncannily computerlike"
- Genes are encoded in strands of DNA called **chromosomes** (23 pairs in a human body)
- **Genotype** determines **phenotype** via a complex mapping (not one to one)

1. https://www.lrb.co.uk/the-paper/v15/n08/steve-jones/a-slower-kind-of-bang

Natural Selection and Genetics

- According to the geneticist Steve Jones[1],

  > *Had [Darwin] published his [book] today he would have been in trouble with the Trades Description Act because if there is one thing which The Origin of Species is not about, it is the origin of species.*

- Darwin didn't know what was the unit of natural selection, namely, **genes**
- The information required to build a living organism is coded in the DNA of that organism (four nucleotides, A, G, T, C, in a double helix spiral)
- Nucleotides encode specific amino acids, out of which all proteins are composed
- R. Dawkins's *River out of Eden*: "[...] genes [...] are long strings of pure digital information. [...] The machine code of genes is uncannily computerlike"
- Genes are encoded in strands of DNA called **chromosomes** (23 pairs in a human body)
- **Genotype** determines **phenotype** via a complex mapping (not one to one)

1. https://www.lrb.co.uk/the-paper/v15/n08/steve-jones/a-slower-kind-of-bang

Natural Selection and Genetics

- According to the geneticist Steve Jones[1],

  > *Had [Darwin] published his [book] today he would have been in trouble with the Trades Description Act because if there is one thing which The Origin of Species is not about, it is the origin of species.*

- Darwin didn't know what was the unit of natural selection, namely, **genes**
- The information required to build a living organism is coded in the DNA of that organism (four nucleotides, A, G, T, C, in a double helix spiral)
- Nucleotides encode specific amino acids, out of which all proteins are composed
- R. Dawkins's *River out of Eden*: "[...] genes [...] are long strings of pure digital information. [...] The machine code of genes is uncannily computerlike"
- Genes are encoded in strands of DNA called **chromosomes** (23 pairs in a human body)
- Genotype determines phenotype via a complex mapping (not one to one)

1. https://www.lrb.co.uk/the-paper/v15/n08/steve-jones/a-slower-kind-of-bang

Natural Selection and Genetics

- According to the geneticist Steve Jones[1],

  > *Had [Darwin] published his [book] today he would have been in trouble with the Trades Description Act because if there is one thing which The Origin of Species is not about, it is the origin of species.*

- Darwin didn't know what was the unit of natural selection, namely, **genes**
- The information required to build a living organism is coded in the DNA of that organism (four nucleotides, A, G, T, C, in a double helix spiral)
- Nucleotides encode specific amino acids, out of which all proteins are composed
- R. Dawkins's *River out of Eden*: "[...] genes [...] are long strings of pure digital information. [...] The machine code of genes is uncannily computerlike"
- Genes are encoded in strands of DNA called **chromosomes** (23 pairs in a human body)
- **Genotype** determines **phenotype** via a complex mapping (not one to one)

1. https://www.lrb.co.uk/the-paper/v15/n08/steve-jones/a-slower-kind-of-bang

- According to the geneticist Steve Jones[1],

  > *Had [Darwin] published his [book] today he would have been in trouble with the Trades Description Act because if there is one thing which The Origin of Species is not about, it is the origin of species.*

- Darwin didn't know what was the unit of natural selection, namely, **genes**
- The information required to build a living organism is coded in the DNA of that organism (four nucleotides, A, G, T, C, in a double helix spiral)
- Nucleotides encode specific amino acids, out of which all proteins are composed
- R. Dawkins's *River out of Eden*: "[...] genes [...] are long strings of pure digital information. [...] The machine code of genes is uncannily computerlike"
- Genes are encoded in strands of DNA called **chromosomes** (23 pairs in a human body)
- **Genotype** determines **phenotype** via a complex mapping (not one to one)

1. https://www.lrb.co.uk/the-paper/v15/n08/steve-jones/a-slower-kind-of-bang

Natural Selection and Genetics

- According to the geneticist Steve Jones[1],

  > *Had [Darwin] published his [book] today he would have been in trouble with the Trades Description Act because if there is one thing which The Origin of Species is not about, it is the origin of species.*

- Darwin didn't know what was the unit of natural selection, namely, **genes**
- The information required to build a living organism is coded in the DNA of that organism (four nucleotides, A, G, T, C, in a double helix spiral)
- Nucleotides encode specific amino acids, out of which all proteins are composed
- R. Dawkins's *River out of Eden*: "[...] genes [...] are long strings of pure digital information. [...] The machine code of genes is uncannily computerlike"
- Genes are encoded in strands of DNA called **chromosomes** (23 pairs in a human body)
- **Genotype** determines **phenotype** via a complex mapping (not one to one)
  - Consider analogy to a house blueprint and a cake recipe
  - It is possible to build a house from a blueprint, and draw a blueprint from an already built house
  - It is very difficult to determine the recipe that has been used to bake a cake

1. https://www.lrb.co.uk/the-paper/v15/n08/steve-jones/a-slower-kind-of-bang

Natural Selection and Genetics

- According to the geneticist Steve Jones[1],

  > *Had [Darwin] published his [book] today he would have been in trouble with the Trades Description Act because if there is one thing which The Origin of Species is not about, it is the origin of species.*

- Darwin didn't know what was the unit of natural selection, namely, **genes**
- The information required to build a living organism is coded in the DNA of that organism (four nucleotides, A, G, T, C, in a double helix spiral)
- Nucleotides encode specific amino acids, out of which all proteins are composed
- R. Dawkins's *River out of Eden*: "[...] genes [...] are long strings of pure digital information. [...] The machine code of genes is uncannily computerlike"
- Genes are encoded in strands of DNA called **chromosomes** (23 pairs in a human body)
- **Genotype** determines **phenotype** via a complex mapping (not one to one)
  - Consider analogy to a house blueprint and a cake recipe
  - It is **possible** to build a house from a blueprint **and** draw a blueprint from an already built house
  - It is **not possible** to determine the recipe just by looking at a cake

1. https://www.lrb.co.uk/the-paper/v15/n08/steve-jones/a-slower-kind-of-bang

Natural Selection and Genetics

- According to the geneticist Steve Jones[1],

  > *Had [Darwin] published his [book] today he would have been in trouble with the Trades Description Act because if there is one thing which The Origin of Species is not about, it is the origin of species.*

- Darwin didn't know what was the unit of natural selection, namely, **genes**
- The information required to build a living organism is coded in the DNA of that organism (four nucleotides, A, G, T, C, in a double helix spiral)
- Nucleotides encode specific amino acids, out of which all proteins are composed
- R. Dawkins's *River out of Eden*: "[...] genes [...] are long strings of pure digital information. [...] The machine code of genes is uncannily computerlike"
- Genes are encoded in strands of DNA called **chromosomes** (23 pairs in a human body)
- **Genotype** determines **phenotype** via a complex mapping (not one to one)
  - Consider analogy to a house blueprint and a cake recipe
  - It is **possible** to build a house from a blueprint **and** draw a blueprint from an already built house
  - It is **not possible** to determine the recipe just by looking at a cake

1. https://www.lrb.co.uk/the-paper/v15/n08/steve-jones/a-slower-kind-of-bang

Natural Selection and Genetics

- According to the geneticist Steve Jones[1],

  > *Had [Darwin] published his [book] today he would have been in trouble with the Trades Description Act because if there is one thing which The Origin of Species is not about, it is the origin of species.*

- Darwin didn't know what was the unit of natural selection, namely, **genes**
- The information required to build a living organism is coded in the DNA of that organism (four nucleotides, A, G, T, C, in a double helix spiral)
- Nucleotides encode specific amino acids, out of which all proteins are composed
- R. Dawkins's *River out of Eden*: "[...] genes [...] are long strings of pure digital information. [...] The machine code of genes is uncannily computerlike"
- Genes are encoded in strands of DNA called **chromosomes** (23 pairs in a human body)
- **Genotype** determines **phenotype** via a complex mapping (not one to one)
  - Consider analogy to a house blueprint and a cake recipe
  - It is **possible** to build a house from a blueprint **and** draw a blueprint from an already built house
  - It is **not possible** to determine the recipe just by looking at a cake

1. https://www.lrb.co.uk/the-paper/v15/n08/steve-jones/a-slower-kind-of-bang

Natural Selection and Genetics

- According to the geneticist Steve Jones[1],

  > *Had [Darwin] published his [book] today he would have been in trouble with the Trades Description Act because if there is one thing which The Origin of Species is not about, it is the origin of species.*

- Darwin didn't know what was the unit of natural selection, namely, **genes**
- The information required to build a living organism is coded in the DNA of that organism (four nucleotides, A, G, T, C, in a double helix spiral)
- Nucleotides encode specific amino acids, out of which all proteins are composed
- R. Dawkins's *River out of Eden*: "[...] genes [...] are long strings of pure digital information. [...] The machine code of genes is uncannily computerlike"
- Genes are encoded in strands of DNA called **chromosomes** (23 pairs in a human body)
- **Genotype** determines **phenotype** via a complex mapping (not one to one)
  - Consider analogy to a house blueprint and a cake recipe
  - It is **possible** to build a house from a blueprint **and** draw a blueprint from an already built house
  - It is **not possible** to determine the recipe just by looking at a cake

1. https://www.lrb.co.uk/the-paper/v15/n08/steve-jones/a-slower-kind-of-bang

Genetics and Reproduction

- Gametes (sperm and egg cells) contain 23 individual chromosomes (haploid cells)
- Gametes are formed by a special form of cell splitting called **meiosis**, during which the pairs of chromosome undergo **crossover**
  - Gametes are combined in a process called fertilization
  - New zygote rapidly divides creating many cells with the same genetic contents
  - Although all cells contain the same genes, they will behave differently (development)
- Occasionally some of the genetic material changes very slightly (**mutates**) during the process or reproduction
- This can be, neutral, or advantageous

Genetics and Reproduction

- Gametes (sperm and egg cells) contain 23 individual chromosomes (haploid cells)
- Gametes are formed by a special form of cell splitting called **meiosis**, during which the pairs of chromosome undergo **crossover**
  - Gametes are combined in a process called fertilization
  - New zygote rapidly divides creating many cells with the same genetic contents
  - Although all cells contain the same genes, they will behave differently (ontogenesis)
- Occasionally some of the genetic material changes very slightly (**mutates**) during the process or reproduction
- This can be, neutral, or advantageous

Genetics and Reproduction

- Gametes (sperm and egg cells) contain 23 individual chromosomes (haploid cells)
- Gametes are formed by a special form of cell splitting called **meiosis**, during which the pairs of chromosome undergo **crossover**
  - Gametes are combined in a process called fertilization
  - New zygote rapidly divides creating many cells with the same genetic contents
  - Although all cells contain the same genes, they will behave differently (ontogenesis)
- Occasionally some of the genetic material changes very slightly (**mutates**) during the process or reproduction
- This can be, neutral, or advantageous

Genetics and Reproduction

- Gametes (sperm and egg cells) contain 23 individual chromosomes (haploid cells)
- Gametes are formed by a special form of cell splitting called **meiosis**, during which the pairs of chromosome undergo **crossover**
  - Gametes are combined in a process called fertilization
  - New zygote rapidly divides creating many cells with the same genetic contents
  - Although all cells contain the same genes, they will behave differently (ontogenesis)
- Occasionally some of the genetic material changes very slightly (**mutates**) during the process or reproduction
- This can be, neutral, or advantageous

Genetics and Reproduction

- Gametes (sperm and egg cells) contain 23 individual chromosomes (haploid cells)
- Gametes are formed by a special form of cell splitting called **meiosis**, during which the pairs of chromosome undergo **crossover**
  - Gametes are combined in a process called fertilization
  - New zygote rapidly divides creating many cells with the same genetic contents
  - Although all cells contain the same genes, they will behave differently (ontogenesis)
- Occasionally some of the genetic material changes very slightly (**mutates**) during the process or reproduction
- This can be, neutral, or advantageous

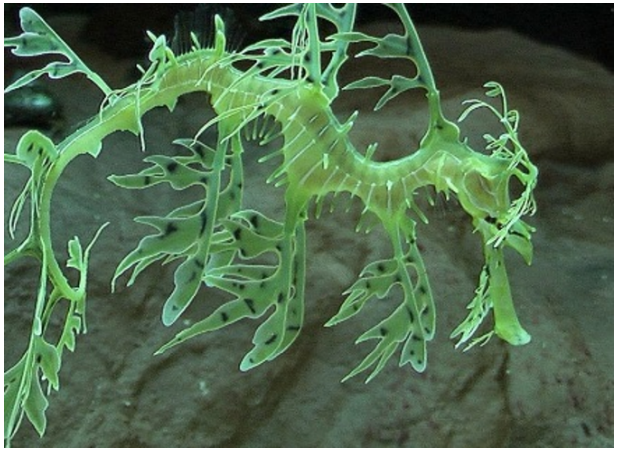
Genetics and Reproduction

- Gametes (sperm and egg cells) contain 23 individual chromosomes (haploid cells)
- Gametes are formed by a special form of cell splitting called **meiosis**, during which the pairs of chromosome undergo **crossover**
  - Gametes are combined in a process called fertilization
  - New zygote rapidly divides creating many cells with the same genetic contents
  - Although all cells contain the same genes, they will behave differently (ontogenesis)
- Occasionally some of the genetic material changes very slightly (**mutates**) during the process or reproduction
- This can be, neutral, or advantageous

Genetics and Reproduction

- Gametes (sperm and egg cells) contain 23 individual chromosomes (haploid cells)
- Gametes are formed by a special form of cell splitting called **meiosis**, during which the pairs of chromosome undergo **crossover**
  - Gametes are combined in a process called fertilization
  - New zygote rapidly divides creating many cells with the same genetic contents
  - Although all cells contain the same genes, they will behave differently (ontogenesis)
- Occasionally some of the genetic material changes very slightly (**mutates**) during the process or reproduction
- This can be, neutral, or advantageous

Some Interesting Examples



Phycodurus eques (leafy sea dragon)

Angraecum sesquipedale (Darwin's orchid) and *Xanthopan morganii* moth

Carausius morosus (stick insect)

Hemeroplanes Triptolemus ("snake caterpillar")

Evolution by Natural Selection Summary

- This is basically the accumulation in a population of heritable changes that allow the species to adapt to its environment
- Accumulation of changes is driven by selective pressure
- Combinations of traits that are better adapted tend to increase representation in population
- Most importantly, this is an **algorithmic process**
- *Watch video*

Evolution by Natural Selection Summary

- This is basically the accumulation in a population of heritable changes that allow the species to adapt to its environment
- Accumulation of changes is driven by selective pressure
- Combinations of traits that are better adapted tend to increase representation in population
- Most importantly, this is an **algorithmic process**
- *Watch video*

Evolution by Natural Selection Summary

- This is basically the accumulation in a population of heritable changes that allow the species to adapt to its environment
- Accumulation of changes is driven by selective pressure
- Combinations of traits that are better adapted tend to increase representation in population
- Most importantly, this is an **algorithmic process**
- *Watch video*

Evolution by Natural Selection Summary

- This is basically the accumulation in a population of heritable changes that allow the species to adapt to its environment
- Accumulation of changes is driven by selective pressure
- Combinations of traits that are better adapted tend to increase representation in population
- Most importantly, this is an **algorithmic process**
- *Watch video*

Evolution by Natural Selection Summary

- This is basically the accumulation in a population of heritable changes that allow the species to adapt to its environment
- Accumulation of changes is driven by selective pressure
- Combinations of traits that are better adapted tend to increase representation in population
- Most importantly, this is an **algorithmic process**
- *Watch video*

The Main Evolutionary Computing Metaphor

- Evolutionary algorithms are **generate-and-test** algorithms
- We can map certain terminology from biology to computer science

Evolution	Problem Solving
Environment	Problem
Individual	Candidate solution
Fitness	Quality of a solution

- Candidate solutions (**individuals**) exist in a **phenotype** space
- They are enconded in **chromosomes**, which exist in a **genotype** space
- Chromosomes contain **genes**, which are located in (usually fixed) positions called **loci** (*sing. locus*) and have a value (**allele**)
- To find the global optimum, every **feasible solution** must be represented in the genotype space

The Main Evolutionary Computing Metaphor

- Evolutionary algorithms are **generate-and-test** algorithms
- We can map certain terminology from biology to computer science

Evolution	Problem Solving
Environment	Problem
Individual	Candidate solution
Fitness	Quality of a solution

- Candidate solutions (**individuals**) exist in a **phenotype** space
- They are enconded in **chromosomes**, which exist in a **genotype** space
- Chromosomes contain **genes**, which are located in (usually fixed) positions called **loci** (*sing. locus*) and have a value (**allele**)
- To find the global optimum, every **feasible solution** must be represented in the genotype space

The Main Evolutionary Computing Metaphor

- Evolutionary algorithms are **generate-and-test** algorithms
- We can map certain terminology from biology to computer science

Evolution	Problem Solving
Environment	Problem
Individual	Candidate solution
Fitness	Quality of a solution

- Candidate solutions (**individuals**) exist in a **phenotype** space
- They are enconded in **chromosomes**, which exist in a **genotype** space
- Chromosomes contain **genes**, which are located in (usually fixed) positions called **loci** (*sing. locus*) and have a value (**allele**)
- To find the global optimum, every **feasible solution** must be represented in the genotype space

The Main Evolutionary Computing Metaphor

- Evolutionary algorithms are **generate-and-test** algorithms
- We can map certain terminology from biology to computer science

Evolution	Problem Solving
Environment	Problem
Individual	Candidate solution
Fitness	Quality of a solution

- Candidate solutions (**individuals**) exist in a **phenotype** space
- They are enconded in **chromosomes**, which exist in a **genotype** space
- Chromosomes contain **genes**, which are located in (usually fixed) positions called **loci** (*sing. locus*) and have a value (**allele**)
- To find the global optimum, **every feasible solution** must be represented in the genotype space

The Main Evolutionary Computing Metaphor

- Evolutionary algorithms are **generate-and-test** algorithms
- We can map certain terminology from biology to computer science

Evolution	Problem Solving
Environment	Problem
Individual	Candidate solution
Fitness	Quality of a solution

- Candidate solutions (**individuals**) exist in a **phenotype** space
- They are enconded in **chromosomes**, which exist in a **genotype** space
- Chromosomes contain **genes**, which are located in (usually fixed) positions called **loci** (*sing. locus*) and have a value (**allele**)
- To find the global optimum, **every feasible solution** must be represented in the genotype space

The Main Evolutionary Computing Metaphor

- Evolutionary algorithms are **generate-and-test** algorithms
- We can map certain terminology from biology to computer science

Evolution	Problem Solving
Environment	Problem
Individual	Candidate solution
Fitness	Quality of a solution

- Candidate solutions (**individuals**) exist in a **phenotype** space
- They are enconded in **chromosomes**, which exist in a **genotype** space
- Chromosomes contain **genes**, which are located in (usually fixed) positions called **loci** (*sing. locus*) and have a value (**allele**)
- To find the global optimum, **every feasible solution** must be represented in the genotype space

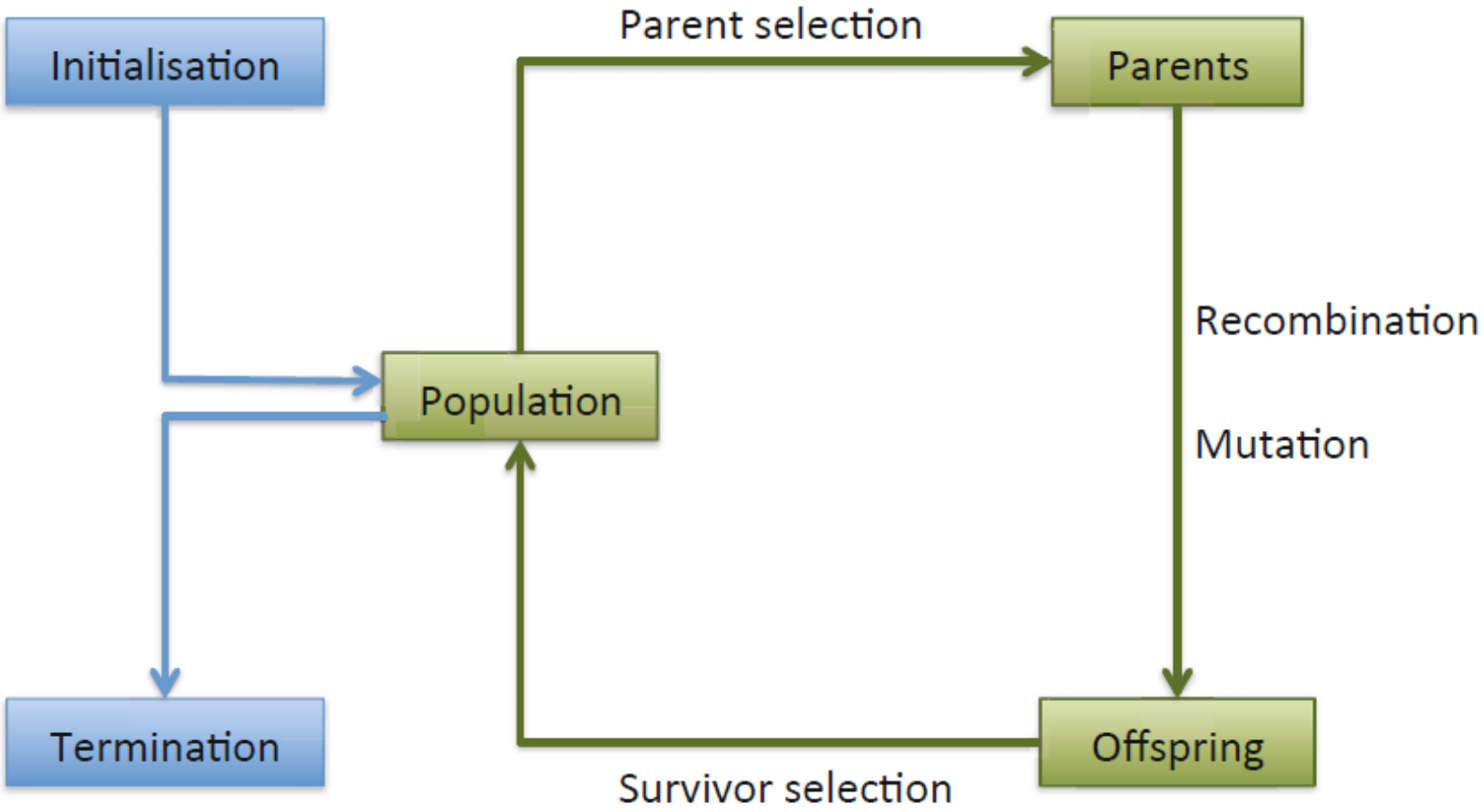
The Main Evolutionary Computing Metaphor

- Evolutionary algorithms are **generate-and-test** algorithms
- We can map certain terminology from biology to computer science

Evolution	Problem Solving
Environment	Problem
Individual	Candidate solution
Fitness	Quality of a solution

- Candidate solutions (**individuals**) exist in a **phenotype** space
- They are enconded in **chromosomes**, which exist in a **genotype** space
- Chromosomes contain **genes**, which are located in (usually fixed) positions called **loci** (*sing. locus*) and have a value (**allele**)
- To find the global optimum, **every feasible solution** must be represented in the genotype space

General Scheme of an Evolutionary Algorithm



Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015), Fig. 3.2

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
- **Parent selection**:
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
- **Parent selection**:
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - For example, if we want to minimize $f(x)$ we set just integers and we present each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example:* if we want to minimize $f(x) = x^2$ over integers and we represent an integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example*: if we want to minimize $f(x) = x^2$ over integers and represent each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example*: if we want to minimize $f(x) = x^2$ over integers and represent each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example*: if we want to minimize $f(x) = x^2$ over integers and represent each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
  - The role is to choose fitter individuals to become parents of the next generation
  - Can be stochastic: lesser fit individuals get a (small) chance of being selected
  - Helps avoid local optima
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example*: if we want to minimize $f(x) = x^2$ over integers and represent each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
  - The role is to choose fitter individuals to become parents of the next generation
  - Can be stochastic: lesser fit individuals get a (small) chance of being selected
  - Helps avoid local optima
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example*: if we want to minimize $f(x) = x^2$ over integers and represent each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
  - The role is to choose fitter individuals to become parents of the next generation
  - Can be stochastic: lesser fit individuals get a (small) chance of being selected
  - Helps avoid local optima
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example*: if we want to minimize $f(x) = x^2$ over integers and represent each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
  - The role is to choose fitter individuals to become parents of the next generation
  - Can be stochastic: lesser fit individuals get a (small) chance of being selected
  - Helps avoid local optima
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example*: if we want to minimize $f(x) = x^2$ over integers and represent each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
  - The role is to choose fitter individuals to become parents of the next generation
  - Can be stochastic: lesser fit individuals get a (small) chance of being selected
  - Helps avoid local optima
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example*: if we want to minimize $f(x) = x^2$ over integers and represent each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
  - The role is to choose fitter individuals to become parents of the next generation
  - Can be stochastic: lesser fit individuals get a (small) chance of being selected
  - Helps avoid local optima
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example*: if we want to minimize $f(x) = x^2$ over integers and represent each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
  - The role is to choose fitter individuals to become parents of the next generation
  - Can be stochastic: lesser fit individuals get a (small) chance of being selected
  - Helps avoid local optima
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example*: if we want to minimize $f(x) = x^2$ over integers and represent each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
  - The role is to choose fitter individuals to become parents of the next generation
  - Can be stochastic: lesser fit individuals get a (small) chance of being selected
  - Helps avoid local optima
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:
  - When the optimum is attained
  - When maximum allowed time/number of fitness evaluations is reached
  - When fitness improvement remains insufficient for successive iterations
  - When the population diversity drops under a critical low level

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example*: if we want to minimize $f(x) = x^2$ over integers and represent each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
  - The role is to choose fitter individuals to become parents of the next generation
  - Can be stochastic: lesser fit individuals get a (small) chance of being selected
  - Helps avoid local optima
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:
  - When the optimum is achieved
  - When maximally allowed time (number of fitness function evaluations) elapses
  - When fitness improvement remains insufficient for some time
  - When the population diversity drops under a critical level

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example*: if we want to minimize $f(x) = x^2$ over integers and represent each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
  - The role is to choose fitter individuals to become parents of the next generation
  - Can be stochastic: lesser fit individuals get a (small) chance of being selected
  - Helps avoid local optima
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:
  - When the optimum is achieved
  - When maximally allowed time (number of fitness function evaluations) elapses
  - When fitness improvement remains low for some time
  - When the population diversity drops under a certain threshold

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example*: if we want to minimize $f(x) = x^2$ over integers and represent each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
  - The role is to choose fitter individuals to become parents of the next generation
  - Can be stochastic: lesser fit individuals get a (small) chance of being selected
  - Helps avoid local optima
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:
  - When the optimum is achieved
  - When maximally allowed time (number of fitness function evaluations) elapses
  - When fitness improvement remains low for some time
  - When the population diversity drops under a certain threshold

General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example*: if we want to minimize $f(x) = x^2$ over integers and represent each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
  - The role is to choose fitter individuals to become parents of the next generation
  - Can be stochastic: lesser fit individuals get a (small) chance of being selected
  - Helps avoid local optima
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:
  - When the optimum is achieved
  - When maximally allowed time (number of fitness function evaluations) elapses
  - When fitness improvement remains low for some time
  - When the population diversity drops under a certain threshold

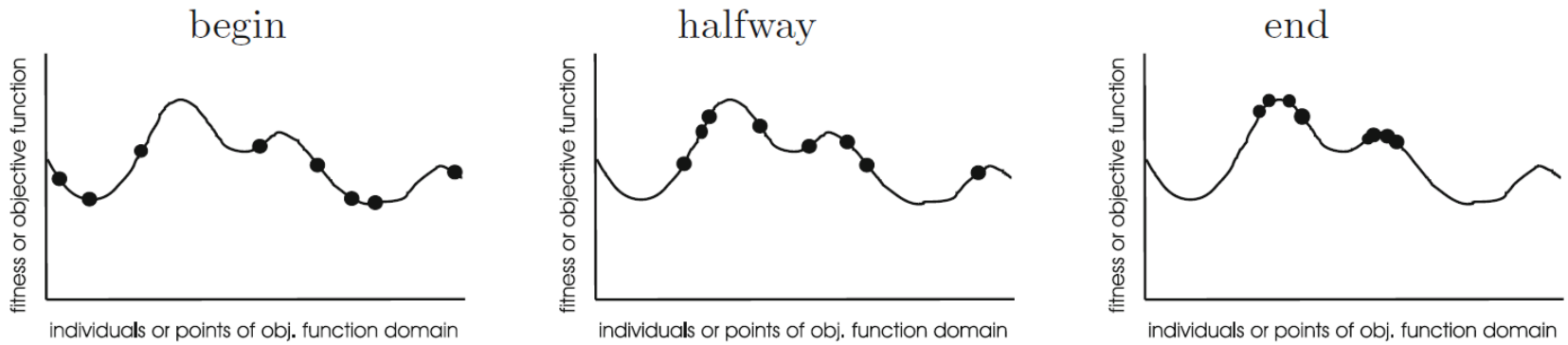
General Scheme of an Evolutionary Algorithm

- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example*: if we want to minimize $f(x) = x^2$ over integers and represent each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
  - The role is to choose fitter individuals to become parents of the next generation
  - Can be stochastic: lesser fit individuals get a (small) chance of being selected
  - Helps avoid local optima
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:
  - When the optimum is achieved
  - When maximally allowed time (number of fitness function evaluations) elapses
  - When fitness improvement remains low for some time
  - When the population diversity drops under a certain threshold

General Scheme of an Evolutionary Algorithm

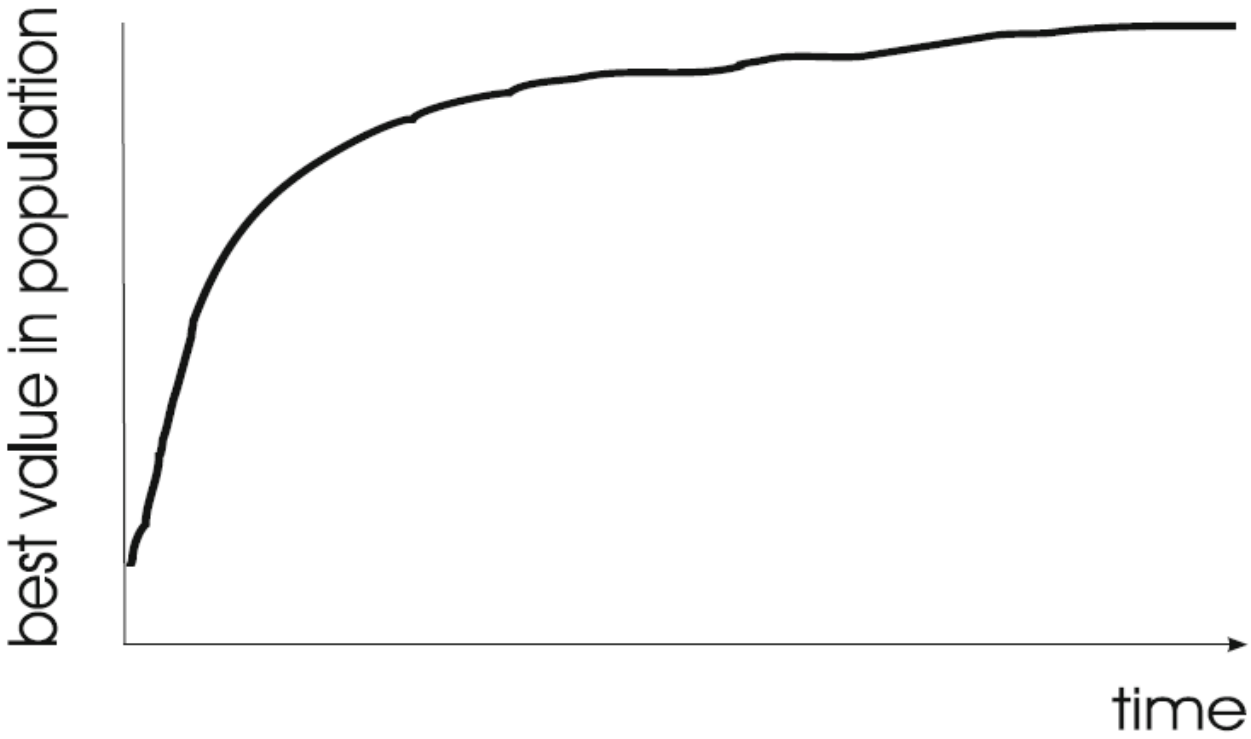
- **Initialization**:
  - The first population (usually populations are of fixed size) is filled with **randomly** generated individuals
  - Initialization needs to ensure even spread and mixture of possible allele values
  - In some cases, initial population can contain existing solutions, or it can be filled with problem-specific heuristics
- **Evaluation** using a **fitness function**:
  - It represents the requirements that the population should adapt to
  - It assigns a single real-valued fitness to each phenotype
  - *Example*: if we want to minimize $f(x) = x^2$ over integers and represent each integer with binary code, then fitness of 10010 would be $18^2 = 324$
- **Parent selection**:
  - The role is to choose fitter individuals to become parents of the next generation
  - Can be stochastic: lesser fit individuals get a (small) chance of being selected
  - Helps avoid local optima
- **Recombination (crossover)**: merges information from parents into offspring thus creating diversity in population
- **Mutation**: (usually stochastically) acts on one genotype and delivers another
- **Survivor selection**: as the population is usually fixed in size, it is necessary to choose the most fit individuals to pass on to a new generation
- **Termination**:
  - When the optimum is achieved
  - When maximally allowed time (number of fitness function evaluations) elapses
  - When fitness improvement remains low for some time
  - When the population diversity drops under a certain threshold

Typical progress of an EA



Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015), Fig. 3.4

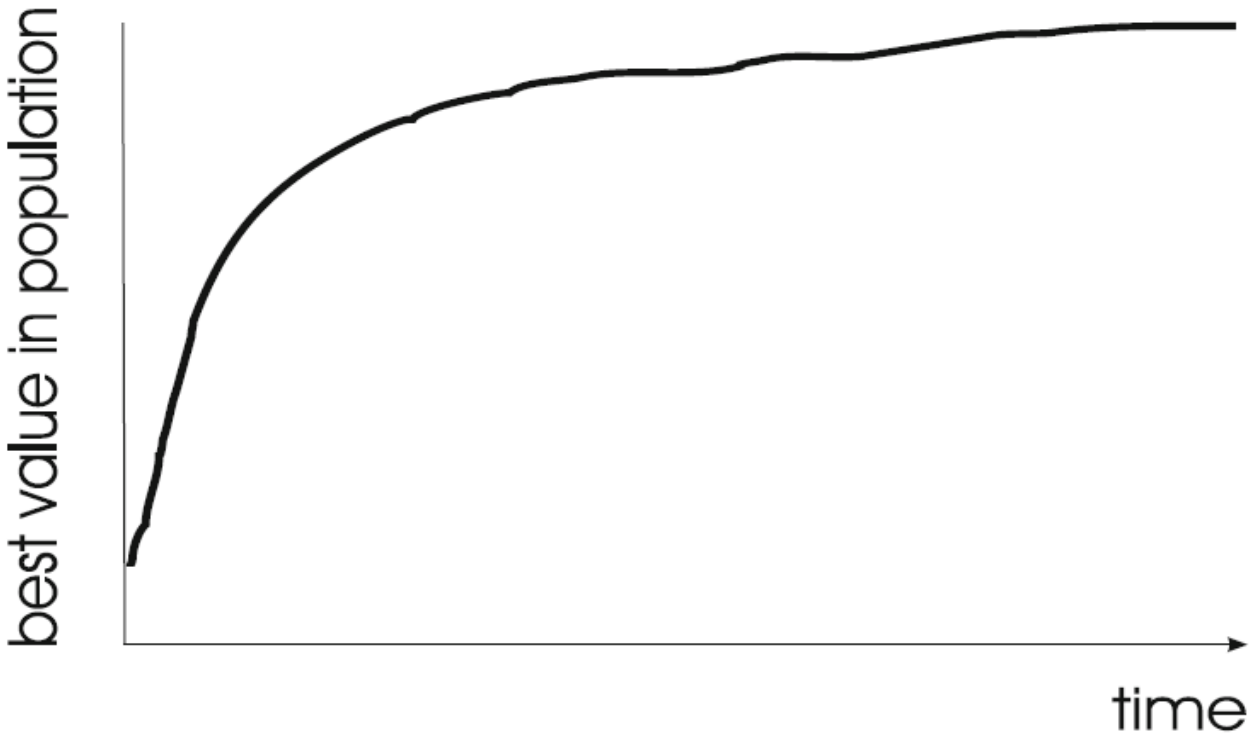
Anytime behavior of an EA



Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015), Fig. 3.5

- Can stop *any time* as soon as we have "good" solutions
- Not necessary to initialize very thoroughly due to rapid fitness increase
- Not necessary to run an algorithm too long due to slow fitness increase

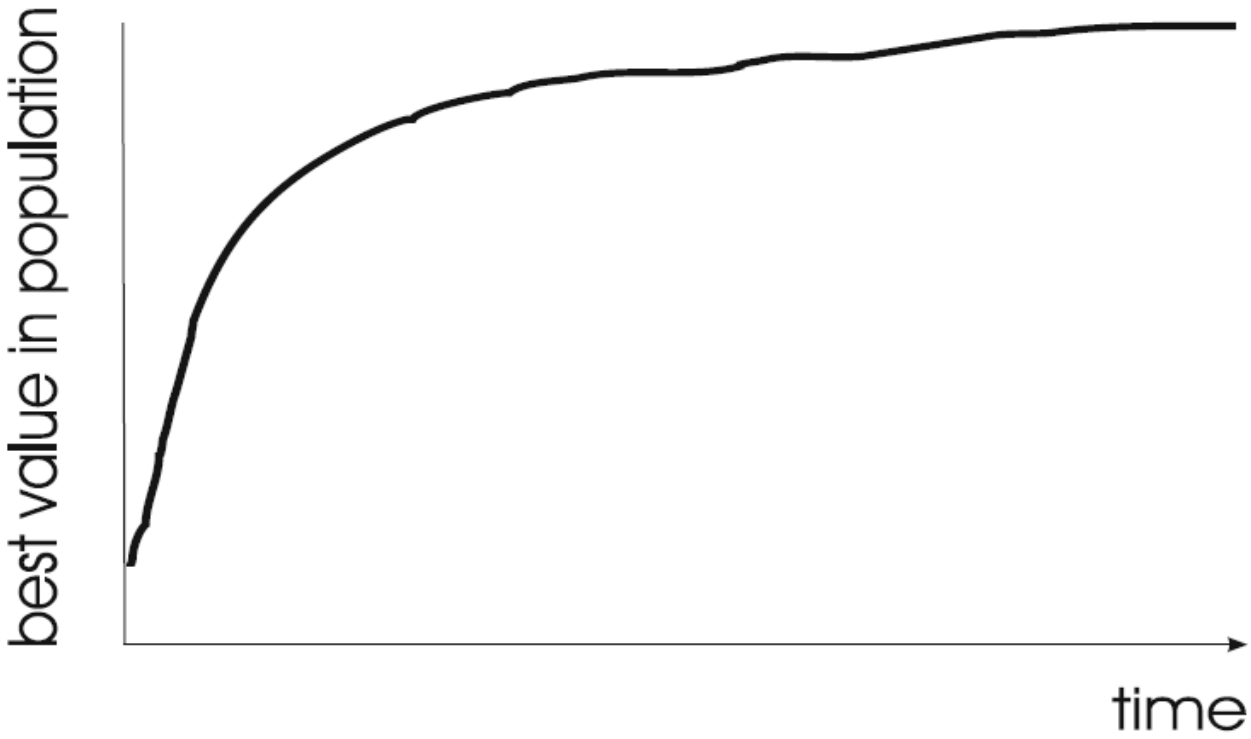
Anytime behavior of an EA



Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015), Fig. 3.5

- Can stop *any time* as soon as we have "good" solutions
- Not necessary to initialize very thoroughly due to rapid fitness increase
- Not necessary to run an algorithm too long due to slow fitness increase

Anytime behavior of an EA



Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015), Fig. 3.5

- Can stop *any time* as soon as we have "good" solutions
- Not necessary to initialize very thoroughly due to rapid fitness increase
- Not necessary to run an algorithm too long due to slow fitness increase

Evolutionary Computing Can Generate New Solutions

Keane, A.: An Introduction to Evolutionary Computing in Design Search and Optimisation. In: Kallel, L., Naudts, B., Rogers, A. (eds.). Theoretical Aspects of Evolutionary Computing. Natural Computing Series. Springer, Berlin, Heidelberg (2001), Figs. 3–4

- Constructing a boom in space: they have severe vibration problems (no air to provide damping)
- The objective was to use an EA to design the geometry of the beam such that the vibrational noise does not travel through it
- Initially a flat two-dimensional beam was considered
- A genetic algorithm was used:

- Improvement was about 3 decades of performance (several thousands of percent!)
- *Watch video*

Evolutionary Computing Can Generate New Solutions

Keane, A.: An Introduction to Evolutionary Computing in Design Search and Optimisation. In: Kallel, L., Naudts, B., Rogers, A. (eds.). Theoretical Aspects of Evolutionary Computing. Natural Computing Series. Springer, Berlin, Heidelberg (2001), Figs. 3–4

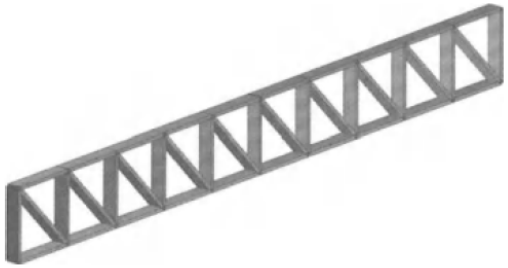
- Constructing a boom in space: they have severe vibration problems (no air to provide damping)
- The objective was to use an EA to design the geometry of the beam such that the vibrational noise does not travel through it
- Initially a flat two-dimensional beam was considered
- A genetic algorithm was used:
  - Population size of 50
  - Was run for 15 generations
  - Took 2 weeks (!) of computation using [illegible] computer workstations
- Improvement was about 3 decades of performance (several thousands of percent!)
- *Watch video*

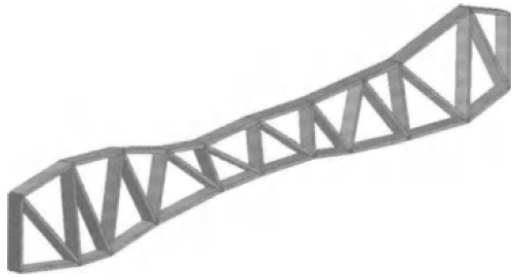
Evolutionary Computing Can Generate New Solutions

Keane, A.: An Introduction to Evolutionary Computing in Design Search and Optimisation. In: Kallel, L., Naudts, B., Rogers, A. (eds.). Theoretical Aspects of Evolutionary Computing. Natural Computing Series. Springer, Berlin, Heidelberg (2001), Figs. 3–4

- Constructing a boom in space: they have severe vibration problems (no air to provide damping)
- The objective was to use an EA to design the geometry of the beam such that the vibrational noise does not travel through it
- Initially a flat two-dimensional beam was considered
- A genetic algorithm was used:
  - Population size of 300
  - Was run for 15 generations
  - It took 2 weeks (!) of computation using [illegible] computer
- Improvement was about 3 decades of performance (several thousands of percent!)
- *Watch video*

Evolutionary Computing Can Generate New Solutions





Keane, A.: An Introduction to Evolutionary Computing in Design Search and Optimisation. In: Kallel, L., Naudts, B., Rogers, A. (eds.). Theoretical Aspects of Evolutionary Computing. Natural Computing Series. Springer, Berlin, Heidelberg (2001), Figs. 3–4

- Constructing a boom in space: they have severe vibration problems (no air to provide damping)
- The objective was to use an EA to design the geometry of the beam such that the vibrational noise does not travel through it
- Initially a flat two-dimensional beam was considered
- A genetic algorithm was used:
  - Population size of 300
  - Was run for 15 generations
  - It took 3 weeks (!) of computation using 11 (!) parallel workstations
- Improvement was about 3 decades of performance (several thousands of percent!)
- *Watch video*

Evolutionary Computing Can Generate New Solutions

Keane, A.: An Introduction to Evolutionary Computing in Design Search and Optimisation. In: Kallel, L., Naudts, B., Rogers, A. (eds.). Theoretical Aspects of Evolutionary Computing. Natural Computing Series. Springer, Berlin, Heidelberg (2001), Figs. 3–4

- Constructing a boom in space: they have severe vibration problems (no air to provide damping)
- The objective was to use an EA to design the geometry of the beam such that the vibrational noise does not travel through it
- Initially a flat two-dimensional beam was considered
- A genetic algorithm was used:
  - Population size of 300
  - Was run for 15 generations
  - It took 3 weeks (!) of computation using 11 (!) parallel workstations
- Improvement was about 3 decades of performance (several thousands of percent!)
- *Watch video*

Evolutionary Computing Can Generate New Solutions

Keane, A.: An Introduction to Evolutionary Computing in Design Search and Optimisation. In: Kallel, L., Naudts, B., Rogers, A. (eds.). Theoretical Aspects of Evolutionary Computing. Natural Computing Series. Springer, Berlin, Heidelberg (2001), Figs. 3–4

- Constructing a boom in space: they have severe vibration problems (no air to provide damping)
- The objective was to use an EA to design the geometry of the beam such that the vibrational noise does not travel through it
- Initially a flat two-dimensional beam was considered
- A genetic algorithm was used:
  - Population size of 300
  - Was run for 15 generations
  - It took 3 weeks (!) of computation using 11 (!) parallel workstations
- Improvement was about 3 decades of performance (several thousands of percent!)
- *Watch video*

Evolutionary Computing Can Generate New Solutions

Keane, A.: An Introduction to Evolutionary Computing in Design Search and Optimisation. In: Kallel, L., Naudts, B., Rogers, A. (eds.). Theoretical Aspects of Evolutionary Computing. Natural Computing Series. Springer, Berlin, Heidelberg (2001), Figs. 3–4

- Constructing a boom in space: they have severe vibration problems (no air to provide damping)
- The objective was to use an EA to design the geometry of the beam such that the vibrational noise does not travel through it
- Initially a flat two-dimensional beam was considered
- A genetic algorithm was used:
  - Population size of 300
  - Was run for 15 generations
  - It took 3 weeks (!) of computation using 11 (!) parallel workstations
- Improvement was about 3 decades of performance (several thousands of percent!)
- *Watch video*

Evolutionary Computing Can Generate New Solutions

Keane, A.: An Introduction to Evolutionary Computing in Design Search and Optimisation. In: Kallel, L., Naudts, B., Rogers, A. (eds.). Theoretical Aspects of Evolutionary Computing. Natural Computing Series. Springer, Berlin, Heidelberg (2001), Figs. 3–4

- Constructing a boom in space: they have severe vibration problems (no air to provide damping)
- The objective was to use an EA to design the geometry of the beam such that the vibrational noise does not travel through it
- Initially a flat two-dimensional beam was considered
- A genetic algorithm was used:
  - Population size of 300
  - Was run for 15 generations
  - It took 3 weeks (!) of computation using 11 (!) parallel workstations
- Improvement was about 3 decades of performance (several thousands of percent!)
- *Watch video*

Evolutionary Computing Can Generate New Solutions

Keane, A.: An Introduction to Evolutionary Computing in Design Search and Optimisation. In: Kallel, L., Naudts, B., Rogers, A. (eds.). Theoretical Aspects of Evolutionary Computing. Natural Computing Series. Springer, Berlin, Heidelberg (2001), Figs. 3–4

- Constructing a boom in space: they have severe vibration problems (no air to provide damping)
- The objective was to use an EA to design the geometry of the beam such that the vibrational noise does not travel through it
- Initially a flat two-dimensional beam was considered
- A genetic algorithm was used:
  - Population size of 300
  - Was run for 15 generations
  - It took 3 weeks (!) of computation using 11 (!) parallel workstations
- Improvement was about 3 decades of performance (several thousands of percent!)
- *Watch video*

Evolutionary Algorithms Vs. Other Techniques

- Can work with a **coding** of a parameter set, not the parameters **themselves**
- Search from a **population of points**, not a single point
- Use **payoff** information, not derivatives or other auxiliary knowledge
- Use **stochastic** transition rules, not deterministic ones
- However, evolutionary computing **is not random search!** (*watch video*)

Evolutionary Algorithms Vs. Other Techniques

- Can work with a **coding** of a parameter set, not the parameters **themselves**
- Search from a **population of points**, not a single point
- Use **payoff** information, not derivatives or other auxiliary knowledge
- Use **stochastic** transition rules, not deterministic ones
- However, evolutionary computing **is not random search!** (*watch video*)

Evolutionary Algorithms Vs. Other Techniques

- Can work with a **coding** of a parameter set, not the parameters **themselves**
- Search from a **population of points**, not a single point
- Use **payoff** information, not derivatives or other auxiliary knowledge
- Use **stochastic** transition rules, not deterministic ones
- However, evolutionary computing **is not random search!** (*watch video*)

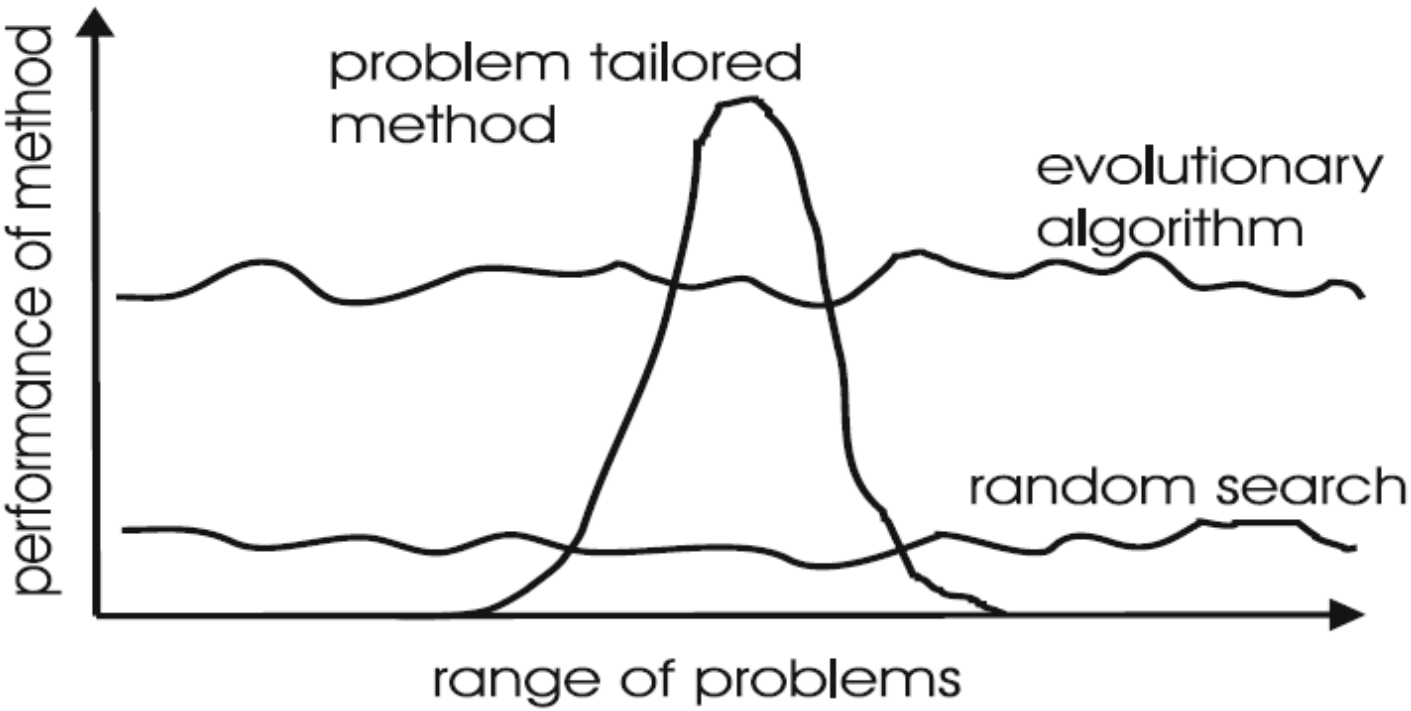
Evolutionary Algorithms Vs. Other Techniques

- Can work with a **coding** of a parameter set, not the parameters **themselves**
- Search from a **population of points**, not a single point
- Use **payoff** information, not derivatives or other auxiliary knowledge
- Use **stochastic** transition rules, not deterministic ones
- However, evolutionary computing **is not random search!** (*watch video*)

Evolutionary Algorithms Vs. Other Techniques

- Can work with a **coding** of a parameter set, not the parameters **themselves**
- Search from a **population of points**, not a single point
- Use **payoff** information, not derivatives or other auxiliary knowledge
- Use **stochastic** transition rules, not deterministic ones
- However, evolutionary computing **is not random search!** (*watch video*)

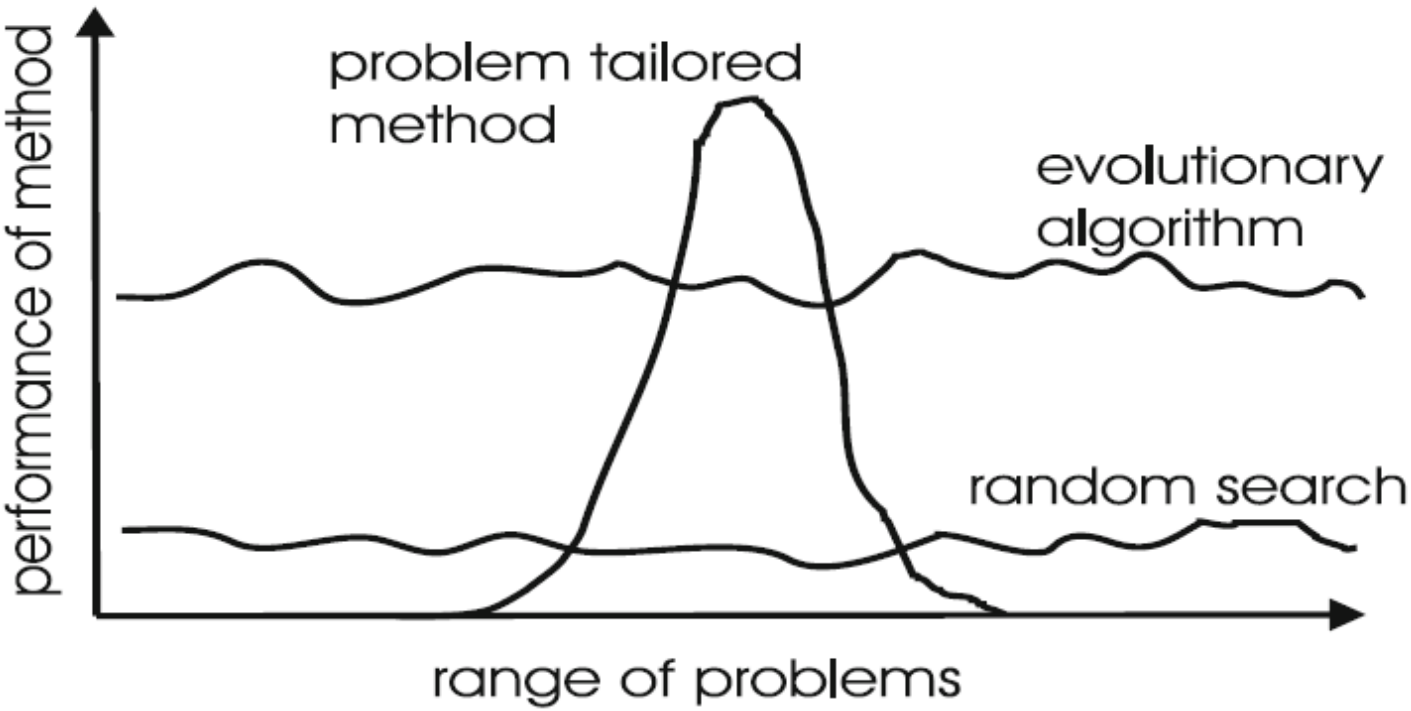
Evolutionary Algorithms Vs. Other Techniques



Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015), Fig. 3.8

- EAs are not limited by assumptions about the search space (such as continuity, derivativity, unimodality, etc.)
- EAs are applicable to big search spaces, not unimodal, not smooth
- Fitness function can be noisy, usually not analytic
- We do not want to spend years looking for the global optimum, but we just want a good sub-optimum in reasonable time

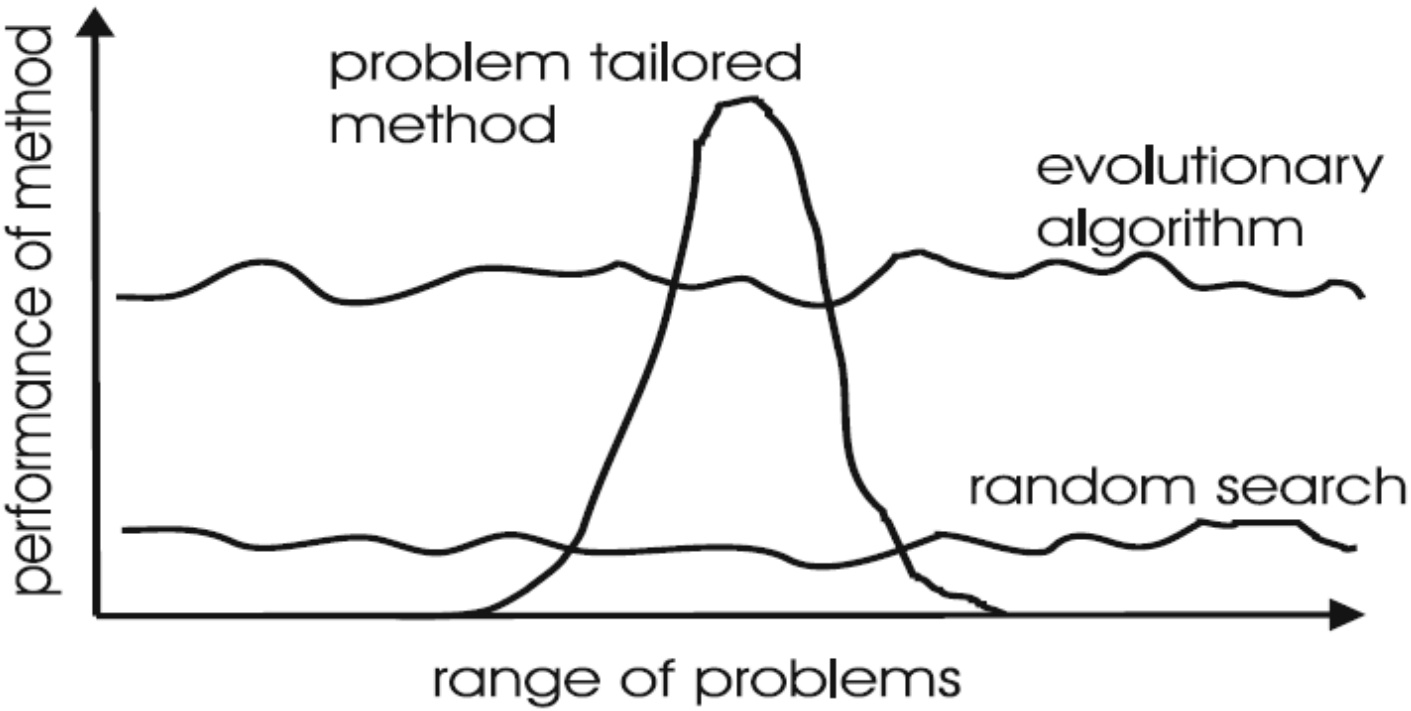
Evolutionary Algorithms Vs. Other Techniques



Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015), Fig. 3.8

- EAs are not limited by assumptions about the search space (such as continuity, derivativity, unimodality, etc.)
- EAs are applicable to big search spaces, not unimodal, not smooth
- Fitness function can be noisy, usually not analytic
- We do not want to spend years looking for the global optimum, but we just want a good sub-optimum in reasonable time

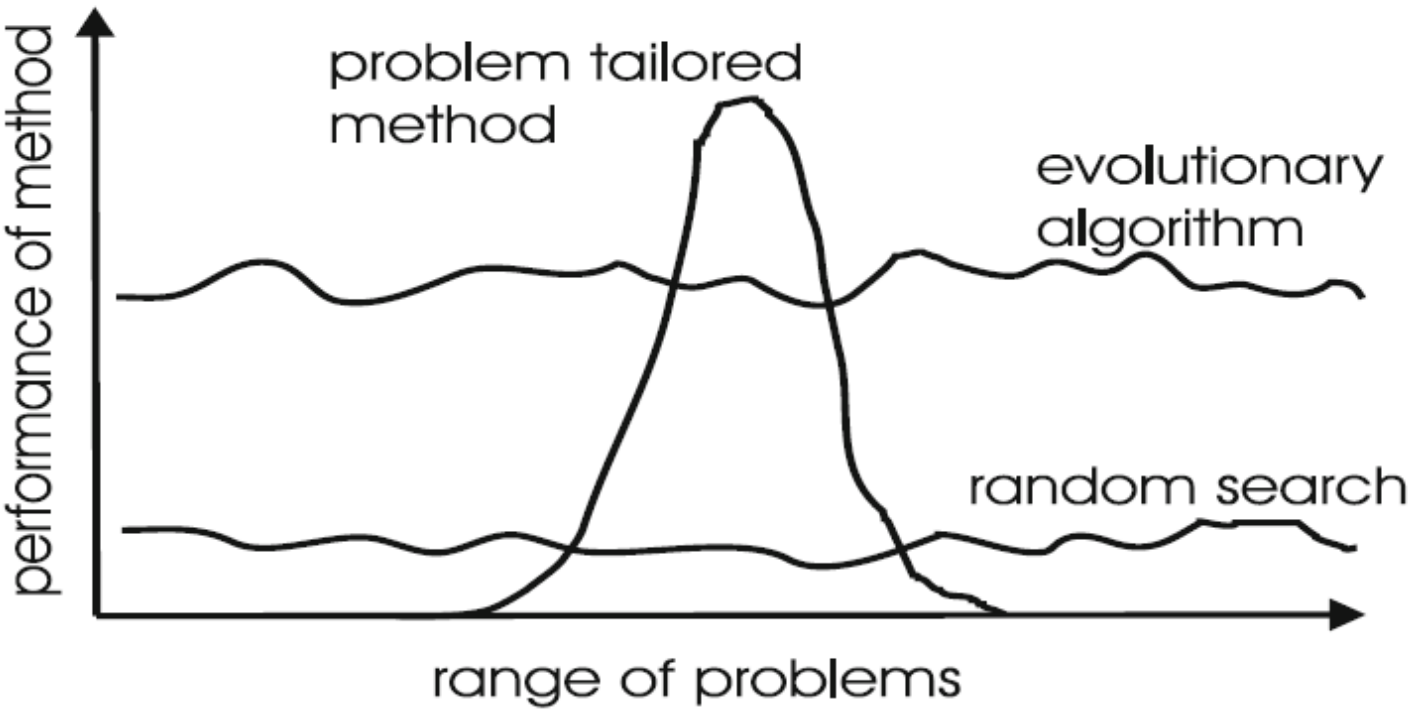
Evolutionary Algorithms Vs. Other Techniques



Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015), Fig. 3.8

- EAs are not limited by assumptions about the search space (such as continuity, derivativity, unimodality, etc.)
- EAs are applicable to big search spaces, not unimodal, not smooth
- Fitness function can be noisy, usually not analytic
- We do not want to spend years looking for the global optimum, but we just want a good sub-optimum in reasonable time

Evolutionary Algorithms Vs. Other Techniques



Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015), Fig. 3.8

- EAs are not limited by assumptions about the search space (such as continuity, derivativity, unimodality, etc.)
- EAs are applicable to big search spaces, not unimodal, not smooth
- Fitness function can be noisy, usually not analytic
- We do not want to spend years looking for the global optimum, but we just want a good sub-optimum in reasonable time

Different Types of EAs

- Historically, different EAs have been associated with different representations
- Strings over a finite alphabet: **genetic algorithms** (Holland, 1975)
- Real-valued vectors: **evolution strategies** (Rechenberg, 1964)
- Finite state machines: **evolutionary programming** (Fogel, Owens and Walsh, 1965)
- Parse trees: **genetic programming** (Koza, 1992)

Different Types of EAs

- Historically, different EAs have been associated with different representations
- Strings over a finite alphabet: **genetic algorithms** (Holland, 1975)
- Real-valued vectors: **evolution strategies** (Rechenberg, 1964)
- Finite state machines: **evolutionary programming** (Fogel, Owens and Walsh, 1965)
- Parse trees: **genetic programming** (Koza, 1992)

Different Types of EAs

- Historically, different EAs have been associated with different representations
- Strings over a finite alphabet: **genetic algorithms** (Holland, 1975)
- Real-valued vectors: **evolution strategies** (Rechenberg, 1964)
- Finite state machines: **evolutionary programming** (Fogel, Owens and Walsh, 1965)
- Parse trees: **genetic programming** (Koza, 1992)

Different Types of EAs

- Historically, different EAs have been associated with different representations
- Strings over a finite alphabet: **genetic algorithms** (Holland, 1975)
- Real-valued vectors: **evolution strategies** (Rechenberg, 1964)
- Finite state machines: **evolutionary programming** (Fogel, Owens and Walsh, 1965)
- Parse trees: **genetic programming** (Koza, 1992)

Different Types of EAs

- Historically, different EAs have been associated with different representations
- Strings over a finite alphabet: **genetic algorithms** (Holland, 1975)
- Real-valued vectors: **evolution strategies** (Rechenberg, 1964)
- Finite state machines: **evolutionary programming** (Fogel, Owens and Walsh, 1965)
- Parse trees: **genetic programming** (Koza, 1992)

Further Reading

- Evolutionary Computing Books:
  - Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015). Chapters 1–3
  - Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Professional (1989). Chapter 1
  - Baeck, T., Fogel, D.B., Michalewicz, Z.: Evolutionary Computation 1. Basic Algorithms and Operators. Taylor & Francis Group LLC (2000). Chapters 1–7
- Popular Science Books:
  - Dawkins, R.: The Selfish Gene. Oxford University Press, Oxford, New York (2006)
  - Dawkins, R.: The Blind Watchmaker. Longmans, London (1986)
  - Dennett, D.C.: Darwin's Dangerous Idea: Evolution and the Meanings of Life. Simon & Schuster (1996)

Outline

Writing an Evolutionary Algorithm

- To implement an evolutionary algorithm **for a specific problem**, it is necessary to make a number of choices:
  - Representation and fitness function
  - Type of crossover applicable to selected representation
  - Type of mutation applicable to selected representation
  - Population model
  - Type of selection
  - Initialization procedure
  - Termination condition(s)
  - Numerical parameters of variation operators (population size, offspring number, crossover probability, mutation probability)
- **Simple genetic algorithms (SGA)** were introduced by John Holland in *Adaptation in Natural and Artifical Systems* (1975)
- Each individual is represented as a **binary string** of **fixed** length l
- It can represent different phenotypes, depending on the problem (binaries, integers, reals)

Writing an Evolutionary Algorithm

- To implement an evolutionary algorithm **for a specific problem**, it is necessary to make a number of choices:
  - Representation and fitness function
  - Type of crossover applicable to selected representation
  - Type of mutation applicable to selected representation
  - Population model
  - Type of selection
  - Initialization procedure
  - Termination condition(s)
  - Numerical parameters of variation operators (population size, offspring number, crossover probability, mutation probability)
- **Simple genetic algorithms (SGA)** were introduced by John Holland in *Adaptation in Natural and Artifical Systems* (1975)
- Each individual is represented as a **binary string** of **fixed** length l
- It can represent different phenotypes, depending on the problem (binaries, integers, reals)

Writing an Evolutionary Algorithm

- To implement an evolutionary algorithm **for a specific problem**, it is necessary to make a number of choices:
  - Representation and fitness function
  - Type of crossover applicable to selected representation
  - Type of mutation applicable to selected representation
  - Population model
  - Type of selection
  - Initialization procedure
  - Termination condition(s)
  - Numerical parameters of variation operators (population size, offspring number, crossover probability, mutation probability)
- **Simple genetic algorithms (SGA)** were introduced by John Holland in *Adaptation in Natural and Artifical Systems* (1975)
- Each individual is represented as a **binary string** of **fixed** length l
- It can represent different phenotypes, depending on the problem (binaries, integers, reals)

Writing an Evolutionary Algorithm

- To implement an evolutionary algorithm **for a specific problem**, it is necessary to make a number of choices:
  - Representation and fitness function
  - Type of crossover applicable to selected representation
  - Type of mutation applicable to selected representation
  - Population model
  - Type of selection
  - Initialization procedure
  - Termination condition(s)
  - Numerical parameters of variation operators (population size, offspring number, crossover probability, mutation probability)

- **Simple genetic algorithms (SGA)** were introduced by John Holland in *Adaptation in Natural and Artifical Systems* (1975)
- Each individual is represented as a **binary string** of **fixed** length l
- It can represent different phenotypes, depending on the problem (binaries, integers, reals)

Writing an Evolutionary Algorithm

- To implement an evolutionary algorithm **for a specific problem**, it is necessary to make a number of choices:
  - Representation and fitness function
  - Type of crossover applicable to selected representation
  - Type of mutation applicable to selected representation
  - Population model
  - Type of selection
  - Initialization procedure
  - Termination condition(s)
  - Numerical parameters of variation operators (population size, offspring number, crossover probability, mutation probability)
- **Simple genetic algorithms (SGA)** were introduced by John Holland in *Adaptation in Natural and Artifical Systems* (1975)
- Each individual is represented as a **binary string** of **fixed** length l
- It can represent different phenotypes, depending on the problem (binaries, integers, reals)

Writing an Evolutionary Algorithm

- To implement an evolutionary algorithm **for a specific problem**, it is necessary to make a number of choices:
  - Representation and fitness function
  - Type of crossover applicable to selected representation
  - Type of mutation applicable to selected representation
  - Population model
  - Type of selection
  - Initialization procedure
  - Termination condition(s)
  - Numerical parameters of variation operators (population size, offspring number, crossover probability, mutation probability)
- **Simple genetic algorithms (SGA)** were introduced by John Holland in *Adaptation in Natural and Artifical Systems* (1975)
- Each individual is represented as a **binary string** of **fixed** length l
- It can represent different phenotypes, depending on the problem (binaries, integers, reals)

Writing an Evolutionary Algorithm

- To implement an evolutionary algorithm **for a specific problem**, it is necessary to make a number of choices:
  - Representation and fitness function
  - Type of crossover applicable to selected representation
  - Type of mutation applicable to selected representation
  - Population model
  - Type of selection
  - Initialization procedure
  - Termination condition(s)
  - Numerical parameters of variation operators (population size, offspring number, crossover probability, mutation probability)
- **Simple genetic algorithms (SGA)** were introduced by John Holland in *Adaptation in Natural and Artifical Systems* (1975)
- Each individual is represented as a **binary string** of **fixed** length l
- It can represent different phenotypes, depending on the problem (binaries, integers, reals)

Writing an Evolutionary Algorithm

- To implement an evolutionary algorithm **for a specific problem**, it is necessary to make a number of choices:
  - Representation and fitness function
  - Type of crossover applicable to selected representation
  - Type of mutation applicable to selected representation
  - Population model
  - Type of selection
  - Initialization procedure
  - Termination condition(s)
  - Numerical parameters of variation operators (population size, offspring number, crossover probability, mutation probability)
- **Simple genetic algorithms (SGA)** were introduced by John Holland in *Adaptation in Natural and Artifical Systems* (1975)
- Each individual is represented as a **binary string** of **fixed** length l
- It can represent different phenotypes, depending on the problem (binaries, integers, reals)

Writing an Evolutionary Algorithm

- To implement an evolutionary algorithm **for a specific problem**, it is necessary to make a number of choices:
  - Representation and fitness function
  - Type of crossover applicable to selected representation
  - Type of mutation applicable to selected representation
  - Population model
  - Type of selection
  - Initialization procedure
  - Termination condition(s)
  - Numerical parameters of variation operators (population size, offspring number, crossover probability, mutation probability)
- **Simple genetic algorithms (SGA)** were introduced by John Holland in *Adaptation in Natural and Artifical Systems* (1975)
- Each individual is represented as a **binary string** of **fixed** length l
- It can represent different phenotypes, depending on the problem (binaries, integers, reals)

Writing an Evolutionary Algorithm

- To implement an evolutionary algorithm **for a specific problem**, it is necessary to make a number of choices:
  - Representation and fitness function
  - Type of crossover applicable to selected representation
  - Type of mutation applicable to selected representation
  - Population model
  - Type of selection
  - Initialization procedure
  - Termination condition(s)
  - Numerical parameters of variation operators (population size, offspring number, crossover probability, mutation probability)
- **Simple genetic algorithms (SGA)** were introduced by John Holland in *Adaptation in Natural and Artifical Systems* (1975)
- Each individual is represented as a **binary string** of **fixed** length l
- It can represent different phenotypes, depending on the problem (binaries, integers, reals)

Writing an Evolutionary Algorithm

- To implement an evolutionary algorithm **for a specific problem**, it is necessary to make a number of choices:
  - Representation and fitness function
  - Type of crossover applicable to selected representation
  - Type of mutation applicable to selected representation
  - Population model
  - Type of selection
  - Initialization procedure
  - Termination condition(s)
  - Numerical parameters of variation operators (population size, offspring number, crossover probability, mutation probability)
- **Simple genetic algorithms (SGA)** were introduced by John Holland in *Adaptation in Natural and Artifical Systems* (1975)
- Each individual is represented as a **binary string** of **fixed** length l
- It can represent different phenotypes, depending on the problem (binaries, integers, reals)

Writing an Evolutionary Algorithm

- To implement an evolutionary algorithm **for a specific problem**, it is necessary to make a number of choices:
  - Representation and fitness function
  - Type of crossover applicable to selected representation
  - Type of mutation applicable to selected representation
  - Population model
  - Type of selection
  - Initialization procedure
  - Termination condition(s)
  - Numerical parameters of variation operators (population size, offspring number, crossover probability, mutation probability)
- **Simple genetic algorithms (SGA)** were introduced by John Holland in *Adaptation in Natural and Artifical Systems* (1975)
- Each individual is represented as a **binary string** of **fixed** length l
- It can represent different phenotypes, depending on the problem (binaries, integers, reals)

Crossover in SGA

- **One-point (simple) crossover**:
  1. Choose a random integer in $[1; l-1]$
  2. Split the parents at this point
  3. Create children by exchanging the tails

Crossover in SGA

- **One-point (simple) crossover**:
  1. Choose a random integer in $[1; l-1]$
  2. Split the parents at this point
  3. Create children by exchanging the tails

Crossover in SGA

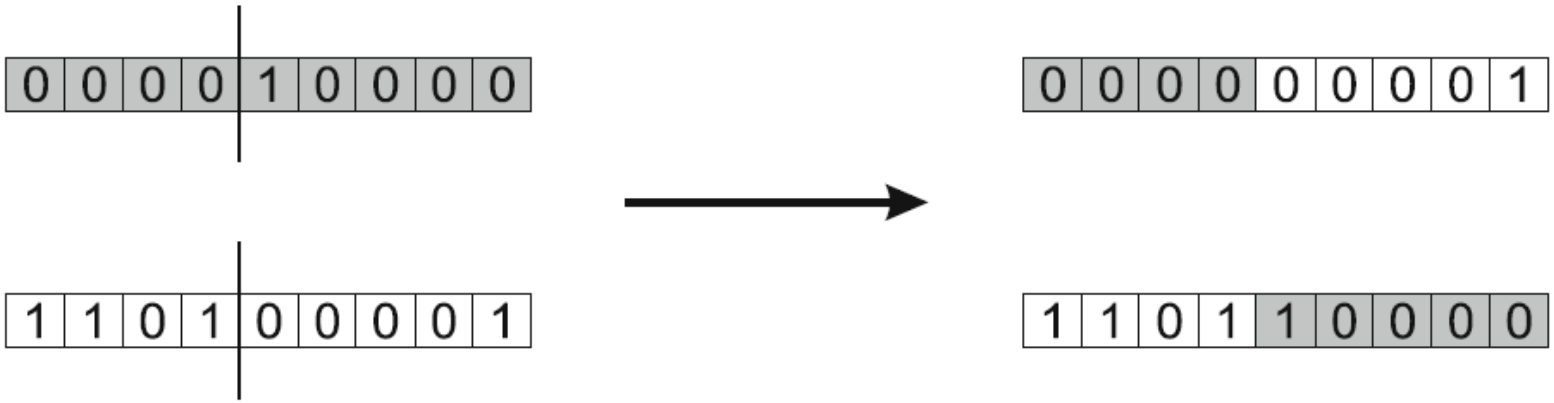
- **One-point (simple) crossover**:
  1. Choose a random integer in $[1; l-1]$
  2. Split the parents at this point
  3. Create children by exchanging the tails

Crossover in SGA

- **One-point (simple) crossover**:
  1. Choose a random integer in $[1; l-1]$
  2. Split the parents at this point
  3. Create children by exchanging the tails

Crossover in SGA

- **One-point (simple) crossover**:
  1. Choose a random integer in $[1; l-1]$
  2. Split the parents at this point
  3. Create children by exchanging the tails



Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015), Fig. 4.2

Mutation in SGA

- **Bit-flipping mutation**: flip each gene independently with a *small* probability p_m

Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin, Heidelberg (2015), Fig. 4.1

Population Models in GA

- There are two basic population models in evolutionary algorithms
- The **generational model**:
  - In each generation, we begin with a population of size μ.
  - We select $\lambda = \mu$ parents.
  - We generate the offspring from this mating pool by means of mutation operators.
  - After each generation, the whole population is replaced by the offspring.
- The **steady-state model**:
- In SGA, the generational model is used

Population Models in GA

- There are two basic population models in evolutionary algorithms
- The **generational model**:
  - In each generation, we begin with a population of size μ
  - We select $\lambda = \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - After each generation, the **whole** population is replaced by the offspring
- The **steady-state model**:
- In SGA, the generational model is used

Population Models in GA

- There are two basic population models in evolutionary algorithms
- The **generational model**:
  - In each generation, we begin with a population of size μ
  - We select $\lambda = \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - After each generation, the **whole** population is replaced by the offspring
- The **steady-state model**:
- In SGA, the generational model is used

Population Models in GA

- There are two basic population models in evolutionary algorithms
- The **generational model**:
  - In each generation, we begin with a population of size μ
  - We select $\lambda = \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - After each generation, the **whole** population is replaced by the offspring
- The **steady-state model**:
- In SGA, the generational model is used

Population Models in GA

- There are two basic population models in evolutionary algorithms
- The **generational model**:
  - In each generation, we begin with a population of size μ
  - We select $\lambda = \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - After each generation, the **whole** population is replaced by the offspring
- The **steady-state model**:
  - In each generation, we begin with a population of size μ
  - We select $\lambda < \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - Only λ individuals from the former population are replaced by the offspring
  - The proportion of the population being replaced in the generation gap is equal to λ/μ
- In SGA, the generational model is used

Population Models in GA

- There are two basic population models in evolutionary algorithms
- The **generational model**:
  - In each generation, we begin with a population of size μ
  - We select $\lambda = \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - After each generation, the **whole** population is replaced by the offspring
- The **steady-state model**:
  - In each generation, ww begin with a population of size μ
  - We select $\lambda < \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - Only λ individuals from the former population are replaced by the offspring
  - The proportion of the population being replaced in the generation gap is equal to λ/μ
- In SGA, the generational model is used

Population Models in GA

- There are two basic population models in evolutionary algorithms
- The **generational model**:
  - In each generation, we begin with a population of size μ
  - We select $\lambda = \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - After each generation, the **whole** population is replaced by the offspring
- The **steady-state model**:
  - In each generation, ww begin with a population of size μ
  - We select $\lambda < \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - Only λ individuals from the initial population are replaced by the offspring
  - The proportion of the population being replaced is the **generational gap**, equal to λ/μ
- In SGA, the generational model is used

Population Models in GA

- There are two basic population models in evolutionary algorithms
- The **generational model**:
  - In each generation, we begin with a population of size μ
  - We select $\lambda = \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - After each generation, the **whole** population is replaced by the offspring
- The **steady-state model**:
  - In each generation, ww begin with a population of size μ
  - We select $\lambda < \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - Only λ individuals from the initial population are replaced by the offspring
  - The proportion of the population being replaced is the **generational gap**, equal to λ/μ
- In SGA, the generational model is used

Population Models in GA

- There are two basic population models in evolutionary algorithms
- The **generational model**:
  - In each generation, we begin with a population of size μ
  - We select $\lambda = \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - After each generation, the **whole** population is replaced by the offspring
- The **steady-state model**:
  - In each generation, ww begin with a population of size μ
  - We select $\lambda < \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - Only λ individuals from the initial population are replaced by the offspring
  - The proportion of the population being replaced is the **generational gap**, equal to λ/μ
- In SGA, the generational model is used

Population Models in GA

- There are two basic population models in evolutionary algorithms
- The **generational model**:
  - In each generation, we begin with a population of size μ
  - We select $\lambda = \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - After each generation, the **whole** population is replaced by the offspring
- The **steady-state model**:
  - In each generation, ww begin with a population of size μ
  - We select $\lambda < \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - Only λ individuals from the initial population are replaced by the offspring
  - The proportion of the population being replaced is the **generational gap**, equal to λ/μ
- In SGA, the generational model is used

Population Models in GA

- There are two basic population models in evolutionary algorithms
- The **generational model**:
  - In each generation, we begin with a population of size μ
  - We select $\lambda = \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - After each generation, the **whole** population is replaced by the offspring
- The **steady-state model**:
  - In each generation, ww begin with a population of size μ
  - We select $\lambda < \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - Only λ individuals from the initial population are replaced by the offspring
  - The proportion of the population being replaced is the **generational gap**, equal to λ/μ
- In SGA, the generational model is used

Population Models in GA

- There are two basic population models in evolutionary algorithms
- The **generational model**:
  - In each generation, we begin with a population of size μ
  - We select $\lambda = \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - After each generation, the **whole** population is replaced by the offspring
- The **steady-state model**:
  - In each generation, ww begin with a population of size μ
  - We select $\lambda < \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - Only λ individuals from the initial population are replaced by the offspring
  - The proportion of the population being replaced is the **generational gap**, equal to λ/μ
- In SGA, the generational model is used

Population Models in GA

- There are two basic population models in evolutionary algorithms
- The **generational model**:
  - In each generation, we begin with a population of size μ
  - We select $\lambda = \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - After each generation, the **whole** population is replaced by the offspring
- The **steady-state model**:
  - In each generation, ww begin with a population of size μ
  - We select $\lambda < \mu$ parents
  - We generate the offspring from this mating pool by means of variation operators
  - Only λ individuals from the initial population are replaced by the offspring
  - The proportion of the population being replaced is the **generational gap**, equal to λ/μ
- In SGA, the generational model is used

Selection in SGA

- In SGA, we use **fitness proportionate selection**
- The probability that individual x_i, $i = 1, \ldots, \mu$, will be selected as a parent is given by

$$p_{x_i} = \frac{f(x_i)}{\sum_{j=1}^{\mu} f(x_j)}$$

- Here, $f(x_i)$ is the fitness of individual x_i
- All fitness values are assumed to be non-negative
- Implementation is done using a *Stochastic Universal Sampling* algorithm[2]:

2. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Grefenstette, J.J. (ed.) Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications, pp. 14–21. Lawrence Erlbaum, Hillsdale, New Jersey (1987)

Selection in SGA

- In SGA, we use **fitness proportionate selection**
- The probability that individual x_i, $i = 1, \ldots, \mu$, will be selected as a parent is given by

$$p_{x_i} = \frac{f(x_i)}{\sum_{j=1}^{\mu} f(x_j)}$$

- Here, $f(x_i)$ is the fitness of individual x_i
- All fitness values are assumed to be non-negative
- Implementation is done using a *Stochastic Universal Sampling* algorithm[2]:

2. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Grefenstette, J.J. (ed.) Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications, pp. 14–21. Lawrence Erlbaum, Hillsdale, New Jersey (1987)

Selection in SGA

- In SGA, we use **fitness proportionate selection**
- The probability that individual x_i, $i = 1, \ldots, \mu$, will be selected as a parent is given by

$$p_{x_i} = \frac{f(x_i)}{\sum_{j=1}^{\mu} f(x_j)}$$

- Here, $f(x_i)$ is the fitness of individual x_i
- All fitness values are assumed to be non-negative
- Implementation is done using a *Stochastic Universal Sampling* algorithm[2]:

2. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Grefenstette, J.J. (ed.) Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications, pp. 14–21. Lawrence Erlbaum, Hillsdale, New Jersey (1987)

Selection in SGA

- In SGA, we use **fitness proportionate selection**
- The probability that individual x_i, $i = 1, \ldots, \mu$, will be selected as a parent is given by

$$p_{x_i} = \frac{f(x_i)}{\sum_{j=1}^{\mu} f(x_j)}$$

- Here, $f(x_i)$ is the fitness of individual x_i
- All fitness values are assumed to be non-negative
- Implementation is done using a *Stochastic Universal Sampling* algorithm[2]:
  - Basically create a biased roulette wheel with each individual having a slot of size, proportional to its fitness
  - There are λ equally spaced wires in the roulette
  - Spin the wheel only once and select the λ individuals that correspond to the λ wires

2. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Grefenstette, J.J. (ed.) Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications, pp. 14–21. Lawrence Erlbaum, Hillsdale, New Jersey (1987)

Selection in SGA

- In SGA, we use **fitness proportionate selection**
- The probability that individual x_i, $i = 1, \ldots, \mu$, will be selected as a parent is given by

$$p_{x_i} = \frac{f(x_i)}{\sum_{j=1}^{\mu} f(x_j)}$$

- Here, $f(x_i)$ is the fitness of individual x_i
- All fitness values are assumed to be non-negative
- Implementation is done using a *Stochastic Universal Sampling* algorithm[2]:
  - Basically create a biased roulette wheel with each individual having a slot of size, proportional to its fitness
  - There are λ *equally spaced* wheel's markers
  - Spin the wheel only once and select the individuals the markers land on
  - Each individual is guaranteed to have either $\lfloor \lambda p_{x_i} \rfloor$ or $\lceil \lambda p_{x_i} \rceil$ offspring

2. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Grefenstette, J.J. (ed.) Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications, pp. 14–21. Lawrence Erlbaum, Hillsdale, New Jersey (1987)

Selection in SGA

- In SGA, we use **fitness proportionate selection**
- The probability that individual x_i, $i = 1, \ldots, \mu$, will be selected as a parent is given by

$$p_{x_i} = \frac{f(x_i)}{\sum_{j=1}^{\mu} f(x_j)}$$

- Here, $f(x_i)$ is the fitness of individual x_i
- All fitness values are assumed to be non-negative
- Implementation is done using a *Stochastic Universal Sampling* algorithm[2]:
  - Basically create a biased roulette wheel with each individual having a slot of size, proportional to its fitness
  - There are λ *equally spaced* wheel's markers
  - Spin the wheel only once and select the individuals the markers land on
  - Each individual is guaranteed to have either $\lfloor \lambda p_{x_i} \rfloor$ or $\lceil \lambda p_{x_i} \rceil$ offspring

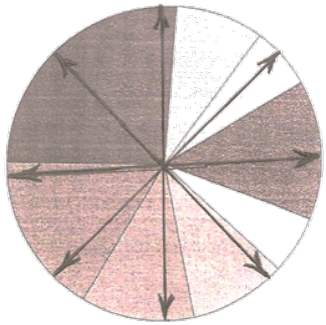
2. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Grefenstette, J.J. (ed.) Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications, pp. 14–21. Lawrence Erlbaum, Hillsdale, New Jersey (1987)

Selection in SGA

- In SGA, we use **fitness proportionate selection**
- The probability that individual x_i, $i = 1, \ldots, \mu$, will be selected as a parent is given by

$$p_{x_i} = \frac{f(x_i)}{\sum_{j=1}^{\mu} f(x_j)}$$

- Here, $f(x_i)$ is the fitness of individual x_i
- All fitness values are assumed to be non-negative
- Implementation is done using a *Stochastic Universal Sampling* algorithm[2]:
  - Basically create a biased roulette wheel with each individual having a slot of size,



    proportional to its fitness
  - There are λ *equally spaced* wheel's markers
  - Spin the wheel only once and select the individuals the markers land on
  - Each individual is guaranteed to have either $\lfloor \lambda p_{x_i} \rfloor$ or $\lceil \lambda p_{x_i} \rceil$ offspring

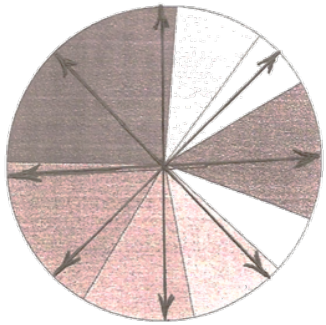
2. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Grefenstette, J.J. (ed.) Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications, pp. 14–21. Lawrence Erlbaum, Hillsdale, New Jersey (1987)

Selection in SGA

- In SGA, we use **fitness proportionate selection**
- The probability that individual x_i, $i = 1, \ldots, \mu$, will be selected as a parent is given by

$$p_{x_i} = \frac{f(x_i)}{\sum_{j=1}^{\mu} f(x_j)}$$

- Here, $f(x_i)$ is the fitness of individual x_i
- All fitness values are assumed to be non-negative
- Implementation is done using a *Stochastic Universal Sampling* algorithm[2]:
  - Basically create a biased roulette wheel with each individual having a slot of size,



    proportional to its fitness
  - There are λ *equally spaced* wheel's markers
  - Spin the wheel only once and select the individuals the markers land on
  - Each individual is guaranteed to have either $\lfloor \lambda p_{x_i} \rfloor$ or $\lceil \lambda p_{x_i} \rceil$ offspring

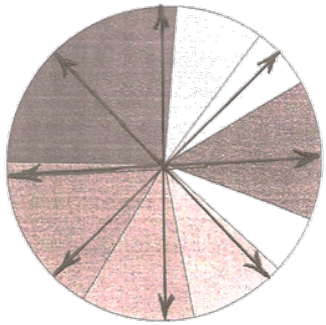
2. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Grefenstette, J.J. (ed.) Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications, pp. 14–21. Lawrence Erlbaum, Hillsdale, New Jersey (1987)

Selection in SGA

- In SGA, we use **fitness proportionate selection**
- The probability that individual x_i, $i = 1, \ldots, \mu$, will be selected as a parent is given by

$$p_{x_i} = \frac{f(x_i)}{\sum_{j=1}^{\mu} f(x_j)}$$

- Here, $f(x_i)$ is the fitness of individual x_i
- All fitness values are assumed to be non-negative
- Implementation is done using a *Stochastic Universal Sampling* algorithm[2]:
  - Basically create a biased roulette wheel with each individual having a slot of size,



    proportional to its fitness
  - There are λ *equally spaced* wheel's markers
  - Spin the wheel only once and select the individuals the markers land on
  - Each individual is guaranteed to have either $\lfloor \lambda p_{x_i} \rfloor$ or $\lceil \lambda p_{x_i} \rceil$ offspring

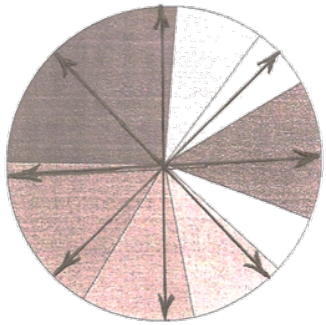
2. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Grefenstette, J.J. (ed.) Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications, pp. 14–21. Lawrence Erlbaum, Hillsdale, New Jersey (1987)

Selection in SGA

- In SGA, we use **fitness proportionate selection**
- The probability that individual x_i, $i = 1, \ldots, \mu$, will be selected as a parent is given by

$$p_{x_i} = \frac{f(x_i)}{\sum_{j=1}^{\mu} f(x_j)}$$

- Here, $f(x_i)$ is the fitness of individual x_i
- All fitness values are assumed to be non-negative
- Implementation is done using a *Stochastic Universal Sampling* algorithm[2]:
  - Basically create a biased roulette wheel with each individual having a slot of size,



    proportional to its fitness
  - There are λ *equally spaced* wheel's markers
  - Spin the wheel only once and select the individuals the markers land on
  - Each individual is guaranteed to have either $\lfloor \lambda p_{x_i} \rfloor$ or $\lceil \lambda p_{x_i} \rceil$ offspring

2. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Grefenstette, J.J. (ed.) Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications, pp. 14–21. Lawrence Erlbaum, Hillsdale, New Jersey (1987)

Other Features of SGA

- **Initial** population is generated **randomly**
- SGA **terminates** when either
  - The solution has been found
  - Maximally allowed number of generations elapsed
- Some suggestions as for numerical parameters[3]
- **Population size** μ should be *moderate* (30–100, depending on the problem)
- **Crossover probability** p_c should by *high* (0.6–1.0) (otherwise the algorithm will be close to random)
- **Mutation probability** p_m should be *low* (about 0.001) (otherwise the algorithm will not converge)

3. De Jong, K.A.: An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)

Other Features of SGA

- **Initial** population is generated **randomly**
- SGA **terminates** when either
  - The solution has been found
  - Maximally allowed number of generations elapsed
- Some suggestions as for numerical parameters[3]
- **Population size** μ should be *moderate* (30–100, depending on the problem)
- **Crossover probability** p_c should by *high* (0.6–1.0) (otherwise the algorithm will be close to random)
- **Mutation probability** p_m should be *low* (about 0.001) (otherwise the algorithm will not converge)

3. De Jong, K.A.: An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)

Other Features of SGA

- **Initial** population is generated **randomly**
- SGA **terminates** when either
  - The solution has been found
  - Maximally allowed number of generations elapsed
- Some suggestions as for numerical parameters[3]
- **Population size** μ should be *moderate* (30–100, depending on the problem)
- **Crossover probability** p_c should by *high* (0.6–1.0) (otherwise the algorithm will be close to random)
- **Mutation probability** p_m should be *low* (about 0.001) (otherwise the algorithm will not converge)

3. De Jong, K.A.: An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)

Other Features of SGA

- **Initial** population is generated **randomly**
- SGA **terminates** when either
  - The solution has been found
  - Maximally allowed number of generations elapsed
- Some suggestions as for numerical parameters[3]
- **Population size** μ should be *moderate* (30–100, depending on the problem):
  - Small populations converge prematurely
  - Large populations exhibit lower performance
- **Crossover probability** p_c should by *high* (0.6–1.0) (otherwise the algorithm will be close to random)
- **Mutation probability** p_m should be *low* (about 0.001) (otherwise the algorithm will not converge)

3. De Jong, K.A.: An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)

Other Features of SGA

- **Initial** population is generated **randomly**
- SGA **terminates** when either
  - The solution has been found
  - Maximally allowed number of generations elapsed
- Some suggestions as for numerical parameters[3]
- **Population size** μ should be *moderate* (30–100, depending on the problem):
  - Small populations converge prematurely
  - Large populations exhibit lower performance
- **Crossover probability** p_c should by *high* (0.6–1.0) (otherwise the algorithm will be close to random)
- **Mutation probability** p_m should be *low* (about 0.001) (otherwise the algorithm will not converge)

3. De Jong, K.A.: An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)

Other Features of SGA

- **Initial** population is generated **randomly**
- SGA **terminates** when either
  - The solution has been found
  - Maximally allowed number of generations elapsed
- Some suggestions as for numerical parameters[3]
- **Population size** μ should be *moderate* (30–100, depending on the problem):
  - Small populations converge prematurely
  - Large populations exhibit lower performance
- **Crossover probability** p_c should by *high* (0.6–1.0) (otherwise the algorithm will be close to random)
- **Mutation probability** p_m should be *low* (about 0.001) (otherwise the algorithm will not converge)

3. De Jong, K.A.: An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)

Other Features of SGA

- **Initial** population is generated **randomly**
- SGA **terminates** when either
  - The solution has been found
  - Maximally allowed number of generations elapsed
- Some suggestions as for numerical parameters[3]
- **Population size** μ should be *moderate* (30–100, depending on the problem):
  - Small populations converge prematurely
  - Large populations exhibit lower performance
- **Crossover probability** p_c should by *high* (0.6–1.0) (otherwise the algorithm will be close to random)
- **Mutation probability** p_m should be *low* (about 0.001) (otherwise the algorithm will not converge)

3. De Jong, K.A.: An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)

Other Features of SGA

- **Initial** population is generated **randomly**
- SGA **terminates** when either
  - The solution has been found
  - Maximally allowed number of generations elapsed
- Some suggestions as for numerical parameters[3]
- **Population size** μ should be *moderate* (30–100, depending on the problem):
  - Small populations converge prematurely
  - Large populations exhibit lower performance
- **Crossover probability** p_c should by *high* (0.6–1.0) (otherwise the algorithm will be close to random)
- **Mutation probability** p_m should be *low* (about 0.001) (otherwise the algorithm will not converge)

3. De Jong, K.A.: An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)

Other Features of SGA

- **Initial** population is generated **randomly**
- SGA **terminates** when either
  - The solution has been found
  - Maximally allowed number of generations elapsed
- Some suggestions as for numerical parameters[3]
- **Population size** μ should be *moderate* (30–100, depending on the problem):
  - Small populations converge prematurely
  - Large populations exhibit lower performance
- **Crossover probability** p_c should by *high* (0.6–1.0) (otherwise the algorithm will be close to random)
- **Mutation probability** p_m should be *low* (about 0.001) (otherwise the algorithm will not converge)

3. De Jong, K.A.: An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)

Other Features of SGA

- **Initial** population is generated **randomly**
- SGA **terminates** when either
  - The solution has been found
  - Maximally allowed number of generations elapsed
- Some suggestions as for numerical parameters[3]
- **Population size** μ should be *moderate* (30–100, depending on the problem):
  - Small populations converge prematurely
  - Large populations exhibit lower performance
- **Crossover probability** p_c should by *high* (0.6–1.0) (otherwise the algorithm will be close to random)
- **Mutation probability** p_m should be *low* (about 0.001) (otherwise the algorithm will not converge)

3. De Jong, K.A.: An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)

Role of Selection, Crossover, and Mutation

- Individual strings in the population can be compared to **ideas**, and their substrings can be compared to **notions**[4]
- Then, Then, genetic algorithms exploit these ideas by:
  - Selecting high quality notions according to their fitness
  - Crossing these notions with other high fitness notions from other strings
- Mutation plays *secondary* role, but is needed to protect against irrecoverable loss of certain gene values

4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Role of Selection, Crossover, and Mutation

- Individual strings in the population can be compared to **ideas**, and their substrings can be compared to **notions**[4]
- Then, Then, genetic algorithms exploit these ideas by:
  - Selecting high quality notions according to their fitness
  - Crossing these notions with other high fitness notions from other strings
- Mutation plays *secondary* role, but is needed to protect against irrecoverable loss of certain gene values

4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Role of Selection, Crossover, and Mutation

- Individual strings in the population can be compared to **ideas**, and their substrings can be compared to **notions**[4]
- Then, Then, genetic algorithms exploit these ideas by:
  - Selecting high quality notions according to their fitness
  - Crossing these notions with other high fitness notions from other strings
- Mutation plays *secondary* role, but is needed to protect against irrecoverable loss of certain gene values

4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Role of Selection, Crossover, and Mutation

- Individual strings in the population can be compared to **ideas**, and their substrings can be compared to **notions**[4]
- Then, Then, genetic algorithms exploit these ideas by:
  - Selecting high quality notions according to their fitness
  - Crossing these notions with other high fitness notions from other strings
- Mutation plays *secondary* role, but is needed to protect against irrecoverable loss of certain gene values

4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Role of Selection, Crossover, and Mutation

- Individual strings in the population can be compared to **ideas**, and their substrings can be compared to **notions**[4]
- Then, Then, genetic algorithms exploit these ideas by:
  - Selecting high quality notions according to their fitness
  - Crossing these notions with other high fitness notions from other strings
- Mutation plays *secondary* role, but is needed to protect against irrecoverable loss of certain gene values

4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Computer Simulation of SGA

Simple Genetic Algorithm Jupyter Notebook, section 1

Premature Convergence

- What we see in the harder version of the problem is *premature convergence*
- There are two main reasons for why it occurs[5]
- At the beginning, it is common to have a few extraordinary individuals in a population of mediocre colleagues
- Such individuals quickly take over the population
- But later on in the run, the population *average* fitness may be close to the population *best* fitness
- And so average individuals and very good individuals get *nearly the same number of offspring*

5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Premature Convergence

- What we see in the harder version of the problem is *premature convergence*
- There are two main reasons for why it occurs[5]
- At the beginning, it is common to have a few extraordinary individuals in a population of mediocre colleagues
- Such individuals quickly take over the population
- But later on in the run, the population *average* fitness may be close to the population *best* fitness
- And so average individuals and very good individuals get *nearly the same number of offspring*

5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Premature Convergence

- What we see in the harder version of the problem is *premature convergence*
- There are two main reasons for why it occurs[5]
- At the beginning, it is common to have a few extraordinary individuals in a population of mediocre colleagues
- Such individuals quickly take over the population
- But later on in the run, the population *average* fitness may be close to the population *best* fitness
- And so average individuals and very good individuals get *nearly the same number of offspring*

5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Premature Convergence

- What we see in the harder version of the problem is *premature convergence*
- There are two main reasons for why it occurs[5]
- At the beginning, it is common to have a few extraordinary individuals in a population of mediocre colleagues
- Such individuals quickly take over the population
- But later on in the run, the population *average* fitness may be close to the population *best* fitness
- And so average individuals and very good individuals get *nearly the same number of offspring*

5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Premature Convergence

- What we see in the harder version of the problem is *premature convergence*
- There are two main reasons for why it occurs[5]
- At the beginning, it is common to have a few extraordinary individuals in a population of mediocre colleagues
- Such individuals quickly take over the population
- But later on in the run, the population *average* fitness may be close to the population *best* fitness
- And so average individuals and very good individuals get *nearly the same number of offspring*

5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Premature Convergence

- What we see in the harder version of the problem is *premature convergence*
- There are two main reasons for why it occurs[5]
- At the beginning, it is common to have a few extraordinary individuals in a population of mediocre colleagues
- Such individuals quickly take over the population
- But later on in the run, the population *average* fitness may be close to the population *best* fitness
- And so average individuals and very good individuals get *nearly the same number of offspring*

5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Fitness scaling

- To overcome these difficulties, **fitness scaling** is commonly used
- We will focus on *one* way of doing this called **linear fitness scaling**
- We apply linear transformation to fitness function $f(x)$ to get $g(x)$ as follows:

$$g(x) = af(x) + b$$

- Coefficients a and b have to satisfy the following constraints:

$$\frac{1}{\mu}\sum_{i=1}^{\mu} g(x_i) = \frac{1}{\mu}\sum_{i=1}^{\mu} f(x_i)$$

$$\max_{i=1,\ldots,\mu} g(x_i) = C_m \max_{i=1,\ldots,\mu} f(x_i)$$

- In practice, it is suggested[6] to use $C_m \in [1.2; 2.0]$
- If, however, application of this formula leads to negative fitness values, the following two conditions need to be satisfied:

$$\frac{1}{\mu}\sum_{i=1}^{\mu} g(x_i) = \frac{1}{\mu}\sum_{i=1}^{\mu} f(x_i)$$

$$\min_{i=1,\ldots,\mu} g(x_i) = 0$$

6. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Fitness scaling

- To overcome these difficulties, **fitness scaling** is commonly used
- We will focus on *one* way of doing this called **linear fitness scaling**
- We apply linear transformation to fitness function $f(x)$ to get $g(x)$ as follows:

$$g(x) = af(x) + b$$

- Coefficients a and b have to satisfy the following constraints:

$$\frac{1}{\mu}\sum_{i=1}^{\mu} g(x_i) = \frac{1}{\mu}\sum_{i=1}^{\mu} f(x_i)$$

$$\max_{i=1,\ldots,\mu} g(x_i) = C_m \max_{i=1,\ldots,\mu} f(x_i)$$

- In practice, it is suggested[6] to use $C_m \in [1.2; 2.0]$
- If, however, application of this formula leads to negative fitness values, the following two conditions need to be satisfied:

$$\frac{1}{\mu}\sum_{i=1}^{\mu} g(x_i) = \frac{1}{\mu}\sum_{i=1}^{\mu} f(x_i)$$

$$\min_{i=1,\ldots,\mu} g(x_i) = 0$$

6. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Fitness scaling

- To overcome these difficulties, **fitness scaling** is commonly used
- We will focus on *one* way of doing this called **linear fitness scaling**
- We apply linear transformation to fitness function $f(x)$ to get $g(x)$ as follows:

$$g(x) = af(x) + b$$

- Coefficients a and b have to satisfy the following constraints:

$$\frac{1}{\mu}\sum_{i=1}^{\mu} g(x_i) = \frac{1}{\mu}\sum_{i=1}^{\mu} f(x_i)$$

$$\max_{i=1,\ldots,\mu} g(x_i) = C_m \max_{i=1,\ldots,\mu} f(x_i)$$

- In practice, it is suggested[6] to use $C_m \in [1.2; 2.0]$
- If, however, application of this formula leads to negative fitness values, the following two conditions need to be satisfied:

$$\frac{1}{\mu}\sum_{i=1}^{\mu} g(x_i) = \frac{1}{\mu}\sum_{i=1}^{\mu} f(x_i)$$

$$\min_{i=1,\ldots,\mu} g(x_i) = 0$$

6. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Fitness scaling

- To overcome these difficulties, **fitness scaling** is commonly used
- We will focus on *one* way of doing this called **linear fitness scaling**
- We apply linear transformation to fitness function $f(x)$ to get $g(x)$ as follows:

$$g(x) = af(x) + b$$

- Coefficients a and b have to satisfy the following constraints:

$$\frac{1}{\mu}\sum_{i=1}^{\mu} g(x_i) = \frac{1}{\mu}\sum_{i=1}^{\mu} f(x_i)$$

$$\max_{i=1,\ldots,\mu} g(x_i) = C_m \max_{i=1,\ldots,\mu} f(x_i)$$

- In practice, it is suggested[6] to use $C_m \in [1.2; 2.0]$
- If, however, application of this formula leads to negative fitness values, the following two conditions need to be satisfied:

$$\frac{1}{\mu}\sum_{i=1}^{\mu} g(x_i) = \frac{1}{\mu}\sum_{i=1}^{\mu} f(x_i)$$

$$\min_{i=1,\ldots,\mu} g(x_i) = 0$$

6. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Fitness scaling

- To overcome these difficulties, **fitness scaling** is commonly used
- We will focus on *one* way of doing this called **linear fitness scaling**
- We apply linear transformation to fitness function $f(x)$ to get $g(x)$ as follows:

$$g(x) = af(x) + b$$

- Coefficients a and b have to satisfy the following constraints:

$$\frac{1}{\mu}\sum_{i=1}^{\mu} g(x_i) = \frac{1}{\mu}\sum_{i=1}^{\mu} f(x_i)$$

$$\max_{i=1,\ldots,\mu} g(x_i) = C_m \max_{i=1,\ldots,\mu} f(x_i)$$

- In practice, it is suggested[6] to use $C_m \in [1.2; 2.0]$
- If, however, application of this formula leads to negative fitness values, the following two conditions need to be satisfied:

$$\frac{1}{\mu}\sum_{i=1}^{\mu} g(x_i) = \frac{1}{\mu}\sum_{i=1}^{\mu} f(x_i)$$

$$\min_{i=1,\ldots,\mu} g(x_i) = 0$$

6. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Fitness scaling

- To overcome these difficulties, **fitness scaling** is commonly used
- We will focus on *one* way of doing this called **linear fitness scaling**
- We apply linear transformation to fitness function $f(x)$ to get $g(x)$ as follows:

$$g(x) = af(x) + b$$

- Coefficients a and b have to satisfy the following constraints:

$$\frac{1}{\mu}\sum_{i=1}^{\mu} g(x_i) = \frac{1}{\mu}\sum_{i=1}^{\mu} f(x_i)$$

$$\max_{i=1,\ldots,\mu} g(x_i) = C_m \max_{i=1,\ldots,\mu} f(x_i)$$

- In practice, it is suggested[6] to use $C_m \in [1.2; 2.0]$
- If, however, application of this formula leads to negative fitness values, the following two conditions need to be satisfied:

$$\frac{1}{\mu}\sum_{i=1}^{\mu} g(x_i) = \frac{1}{\mu}\sum_{i=1}^{\mu} f(x_i)$$

$$\min_{i=1,\ldots,\mu} g(x_i) = 0$$

6. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley Professional (1989)

Computer Simulation of SGA

Simple Genetic Algorithm Jupyter Notebook, section 2