Genetic Algorithms

CS 419/519

GAs: What are they used for?

Problems can be classified in different ways:

- Black box model
- Search problems
- Optimization vs constraint satisfaction
- NP problems

"Black box" model

Model: an input (solution) evaluator

Input: a proposed solution
Output: result of the computation
(May be broad description rather than specific value.)

"Black box" model

Model

Input
Output

- "Black box" model consists of 3 components
- Two components are known, one is unknown
- Each unknown results in a different problem type
Examples

- **Traveling Salesperson problem:**
  - Input: a sequence of destinations
  - Model: function to sum distances between adjacent destinations
  - Output: total distance traveled

- **University Classroom scheduler:**
  - Input: a schedule assigning classes to classrooms
  - Model: function to determine time conflicts, seat shortages, etc
  - Output: values for number of conflicts, seat delta, etc

- **Eight-queens problem:**
  - Input: arrangement of eight queens on chessboard
  - Model: checker for conflicts
  - Output: indication of conflict or not

Important distinction: problem instance vs. input

- **Problem instance:** what we often think of as an “input” to our algorithms.
- **Input:** (for the purposes of this discussion): a candidate solution to the problem.

   - Example: TSP
     - Problem instance: a set of destinations
     - Input: a sequence of the destinations
   - Example: Classroom scheduling
     - Problem instance: sets of classes and classrooms
     - Input: a schedule that may or may not satisfy all constraints

“Black box” model: Input unknown

**Optimization**

- Model and desired output are known. We seek inputs that maximize or minimize the desired variable(s)

   ![Diagram](Diagram)

   - Examples:
     - Time tables for university, call center, or hospital
     - Design specifications (circuit, probe placement)
     - Traveling salesperson problem (TSP)
     - Eight-queens problem

“Black box” model: Optimisation example 1: university timetabling

**Task: find a timetable**

- Enormously big **search space**
- Timetables must be **good**
- “Good” is defined by a number of competing criteria
  - Courses well distributed
  - Not too many late in the day
- Timetables must be **feasible**
  - Satisfies constraints
    - Enough seats
    - No time conflicts
- Vast majority of search space is **infeasible**
"Black box" model: Optimization example 2: satellite structure

Task: find a design

- Optimized satellite designs for NASA to maximize vibration isolation
- Evolving: design structures
- Fitness: vibration resistance
- Evolutionary "creativity"

"Black box" model: Optimization example 3: 8 queens problem

Task: find an arrangement

- Given an 8-by-8 chessboard and 8 queens
- Place the 8 queens on the chessboard without any conflict
- Two queens conflict if they share same row, column or diagonal
- Can be extended to an n queens problem (n>8)

"Black box" model: Model unknown

Modelling

- We have corresponding sets of inputs & outputs. We seek a model that delivers correct output for every known input

- Examples:
  - Stock market prediction
  - Loan applicant evaluation
  - Facial recognition
  - Autonomous driving
"Black box" model: Model unknown

**Modelling**

- Note: modelling problems can be transformed into optimisation problems
  - Error rate (or success rate) of the model is quantity to be minimized (or maximized)

- Examples
  - Evolutionary machine learning
    - Evolve a neural network that maximizes hit rate of identifications
  - Predicting stock exchange
    - Minimize difference between predicted value and actual value
  - Voice control system for smart homes
    - Minimize error in commands performed

"Black box" model: Modeling example: loan applicant evaluation

- British bank evolved creditability model to predict loan paying behavior of new applicants
  - Evolving: prediction models
  - Fitness: model accuracy on historical data

"Black box" model: Output unknown

**Simulation**

- We have a given model. We seek the outputs that arise under different input conditions
- Often used to answer "what if" questions in evolving dynamic environments

**Examples**

- Evolutionary economics, Artificial Life
- Weather forecast system
- Impact analysis of new tax systems

"Black box" model: Modeling example: stock market prediction

**Two methods**

- Build a time machine
  - DeLoreans are hard to find
- Evolve a predictive model
  - Fitness: difference between actual value of DJIA and predicted value
  - What are the inputs?
"Black box" model: Simulation example: evolving artificial societies

- Simulating trade, economic competition, etc. to calibrate models
- Use models to optimize strategies and policies
- Evolutionary economy
- Survival of the fittest is universal (big/small fish)

"Black box" model: Simulation example 2: cosmology

Simulate the physics beginning at some point in time to test our understanding of the universe

Large number of variables and values requires substantial computational resources

Search problems

- Simulation is different from optimization/modelling
- Optimization/modeling problems search through huge space of possibilities
- Search space: collection of all objects of interest including the desired solution(s)
- Question: how large is the search space for different tours through n destinations?

Problems vs. problem solvers

Important distinction:

- search problems: define search spaces
- problem-solvers: describe how to move through search spaces
Optimization vs. constraint satisfaction

- Objective function: a way of assigning a value to a possible solution that reflects its quality
  - Number of un-checked queens (maximize)
  - Length of a tour visiting given set of destinations (minimize)
- Constraint: binary evaluation telling whether a given requirement holds or not
  - Find a configuration of eight queens on a chessboard such that no two queens check each other
  - Find a tour with minimal length where city X is visited after city Y

Goal:
A solution that:
- "performs well" according to the objective function
- satisfies all constraints

<table>
<thead>
<tr>
<th>Constraints</th>
<th>Objective function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Constrained optimization problem</td>
</tr>
<tr>
<td>No</td>
<td>Free optimization problem</td>
</tr>
</tbody>
</table>

Problem: maximize number of unchecked queens on a chess board
Which category?
Free optimization problem (FOP)
### Optimization vs. constraint satisfaction

<table>
<thead>
<tr>
<th>Constraints</th>
<th>Objective function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Constrained optimization problem</td>
</tr>
<tr>
<td>No</td>
<td>Free optimization problem</td>
</tr>
</tbody>
</table>

Problem: find a configuration of eight queens on a chess board such that no two queens check each other

Which category?
- Constraint satisfaction problem (CSP)

---

<table>
<thead>
<tr>
<th>Constraints</th>
<th>Objective function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Constrained optimization problem</td>
</tr>
<tr>
<td>No</td>
<td>Free optimization problem</td>
</tr>
</tbody>
</table>

Problem: minimize length of tour visiting every destination, in a set of n destinations, exactly once (TSP)

Which category?
- Free optimization problem (FOP)

---

<table>
<thead>
<tr>
<th>Constraints</th>
<th>Objective function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Constrained optimization problem</td>
</tr>
<tr>
<td>No</td>
<td>Free optimization problem</td>
</tr>
</tbody>
</table>

Problem: find a TSP tour with minimal length such that city X is visited after city Y

Which category?
- Constrained optimization problem (COP)

---

### Another classification scheme: P and NP

- So far, we have only looked at classifying problems; we have not discussed problem solvers
- For this new classification scheme, we need the properties of the problem solver – we will classify problems by how easy or hard they are to solve
- Applies to combinatorial optimization problems – these are problems for which the variables are discrete rather than continuous
NP problems:
**Key notions**

- **Problem size**: dimensionality of the problem at hand and number of different values for the problem variables
- **Running-time**: number of operations the algorithm takes to terminate
  - Worst-case as a function of problem size
  - Polynomial, super-polynomial, exponential
- **Problem reduction**: transforming current problem into another via mapping

NP problems:
**Class**

- The ‘difficultness’ of a problem can now be classified:
  - **Class P**: algorithm can solve the problem in polynomial time (worst-case running-time for problem size \( n \) is less than \( F(n) \) for some polynomial formula \( F \))
  - **Class NP**: problem can be solved and any solution can be verified within polynomial time by some other algorithm (\( P \) subset of \( NP \))
  - **Class NP-complete**: problem belongs to class \( NP \) and any other problem in \( NP \) can be reduced to this problem by an algorithm running in polynomial time
  - **Class NP-hard**: problem is at least as hard as any other problem in \( NP \)-complete but solution cannot necessarily be verified within polynomial time

NP problems:
**Difference between classes**

- \( P \) is different from \( NP \)-hard
- Not known whether \( P \) is different from \( NP \)

NP problems:
**Why should you care?**

Let’s say we have a computer with a clock rate defined by the speed of light: \( 3 \times 10^{-24} \) sec (~13 orders of magnitude faster than any modern computer)

What is size of search space for TSP?

How many clock cycles for our computer since the dawn of the universe? ~\( 10^{42} \) or about \( 2^{120} \)

If our computer could evaluate one solution per clock cycle, for a TSP instance with 120 destinations it would take more time than the age of the universe to consider all of them.
NP problems:
Yeah, so?

So, this means that it is not possible to guarantee an optimal solution to even moderately sized instances of hard problems.

Where does this leave us?

**Approximation Algorithms**

* Genetic algorithms are a form of approximation algorithm *

**************************************************

* Genetic algorithms are a form of approximation algorithm *

**************************************************