

# CSE276C - Optimization

Henrik I. Christensen



Computer Science and Engineering  
University of California, San Diego

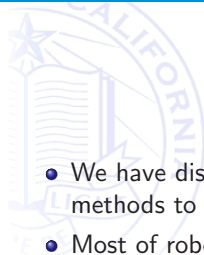
<http://cri.ucsd.edu>

October 2024

# Outline

- 
- 1 Introduction
  - 2 Bracket based methods
  - 3 Downhill Simplex
  - 4 Powell's Method
  - 5 Conjugate Descent/Gradient
  - 6 Stochastic Search
  - 7 Dynamic Programming
  - 8 Summary

# Introduction

- 
- We have discussed approximation and root finding. We can leverage these methods to study optimization.
  - Most of robotics is about optimization
  - Best trajectory between two points
  - Best fit of a model to a swarm of data
  - Optimal coverage of an area for fire monitoring
  - Energy efficient travel from San Diego to Hawaii by water



- Numerical Recipes: Chapter 10
- Numerical Renaissance: Chap 14-16. (Part III)

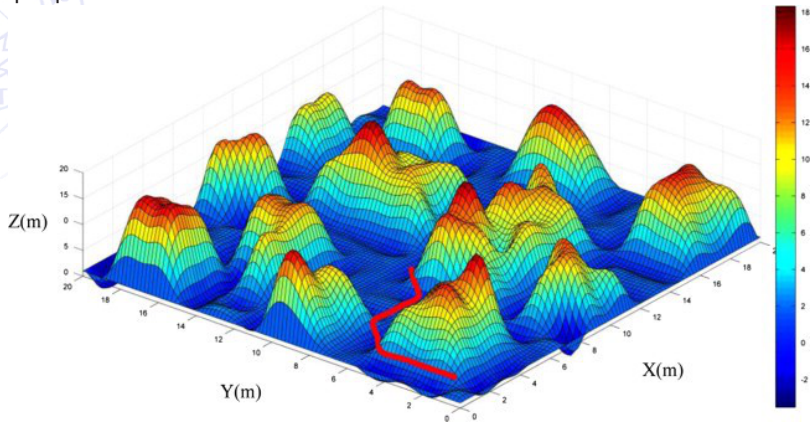
# Example 1



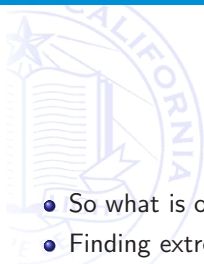
- Optimization of trajectories at high speed

# Path Planning

- Example potential field

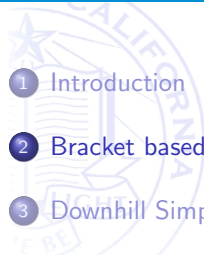


# Optimization

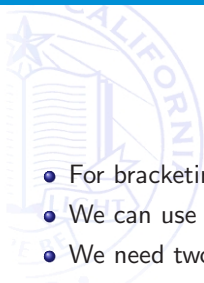


- So what is optimization?
- Finding extrema for a function over a domain
- Minimum or maximum is immaterial as we can use  $f$  or  $-f$
- In many cases we will have local and global extrema
- Consider both deterministic and stochastic approaches

# Outline

- 
- 1 Introduction
  - 2 Bracket based methods
  - 3 Downhill Simplex
  - 4 Powell's Method
  - 5 Conjugate Descent/Gradient
  - 6 Stochastic Search
  - 7 Dynamic Programming
  - 8 Summary

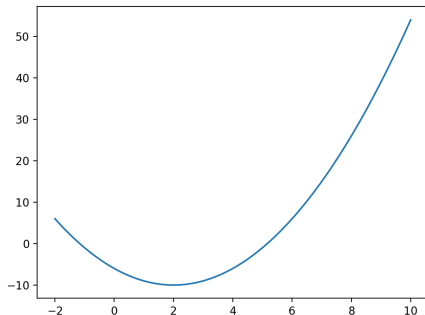




- For bracketing of roots we use bi-section as a basis.
- We can use a similar technique to find an extremum
- We need two points to bracket a root!
- How many points do we need to bracket an extremum?
- We need three points to bracket.
- If we have a triplet  $a < b < c$ . If  $f(b)$  is smaller than  $f(a)$  and  $f(c)$ , then we have a minimum within  $[a, c]$

# Golden Section

- Pick a point between  $(a,b)$  or  $(b,c)$  and evaluate
- Suppose  $x \in (b, c)$  and  $f(x) < f(b)$  then our new triple is  $(b, x, c)$
- Consider the function



- How would you choose a new value of  $x$ ?

# Golden Section (cont.)

- Consider  $(a, b, c)$

$$\frac{b-a}{c-a} = w \qquad \frac{c-b}{c-a} = 1 - w$$

- Lets assume  $x \in (b, c)$  and

$$\frac{x-b}{c-a} = z$$

- The next bracket is then  $w+z$  or  $1-w$
- If we want to make the intervals equal

$$z = 1 - 2w \text{ when } w < \frac{1}{2}$$

- $z$  should be the same distance from  $b$  and  $c$  and  $b$  is from  $a$  and  $c$

$$\frac{z}{1-w} = w$$

- we can rewrite to replace  $z$  and get the equation

$$w^2 - 3w + 1 = 0 \Rightarrow w = \frac{3 - \sqrt{5}}{2} \approx 0.38197$$

- Widely used to select iteration strategies

# Parabolic Interpolation

- We covered Brent's method in root finding and in interpolation
- If we have a triple  $(a, b, c)$  and the values  $f(a)$ ,  $f(b)$ ,  $f(c)$  we can generate a 2nd order interpolation

$$x = b - \frac{1}{2} \frac{(b-a)^2[f(b)-f(c)] - (b-c)^2[f(b)-f(a)]}{(b-a)[f(b)-f(c)] - (b-c)[f(b)-f(a)]}$$

- When would this fail?
- When the triple pair is co-linear!
- The remedy is to use golden section for the co-linear case

# 1-D search w. derivative information

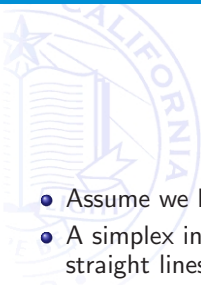


- If we have the triple  $(a, b, c)$  and  $f(a)$ ,  $f(b)$ ,  $f(c)$
- In addition we have  $f'(b)$
- You can use the sign of  $f'(b)$  to choose the next bracket

# Outline

- 
- 1 Introduction
  - 2 Bracket based methods
  - 3 Downhill Simplex
  - 4 Powell's Method
  - 5 Conjugate Descent/Gradient
  - 6 Stochastic Search
  - 7 Dynamic Programming
  - 8 Summary

# Simplex Method



- Assume we have no gradient information or access to formal model.
- A simplex in  $N$  dimensions is composed of  $N+1$  points. Connected by straight lines
  - A 2D simplex is a triangle
  - A 3D simplex is a tetrahedron.
- We have  $N+1$  points  $x_1, \dots, x_{N+1}$

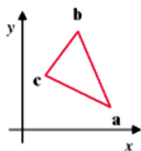
# Downhill Simplex Algorithm

- Initial simple
  - Order the values of the vertices:  $f(x_1) \leq f(x_2) \leq \dots \leq f(x_{N+1})$
- Compute  $x_0$ , the centroid of all points except  $x_{N+1}$
- **Reflection** compute  $x_r = x_0 + \alpha(x_0 - x_{N+1})$ , with  $\alpha > 0$  if the reflection is better than  $f(x_{N+1})$  replace. Restart
- **Expansion** if  $f(x_r) < f(x_1)$  compute  $x_e = x_0 + \gamma(x_r - x_0)$  if  $f(x_e) < f(x_r)$  replace  $x_{N+1}$  else replace  $x_{N+1}$  with  $x_r$ . Restart
- **Contraction** If  $f(x_r) > f(x_N)$  compute  $x_c = x_0 + \rho(x_{N+1} - x_0)$  with  $\rho < .5$ . If  $f(x_c) < f(x_{N+1})$  replace and restart
- **Shrink** Replaces all points except  $x_1$  with  $x_i = x_1 + \sigma(x_i - x_1)$  and restart
- Terminate when update is below a threshold.

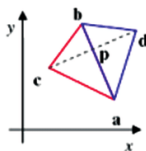


# Simplex illustration

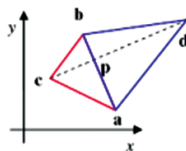
## Downhill Simplex Method



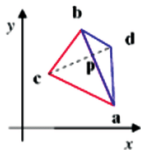
**Initial simplex** with vertices  $a, b, c$ , so that  $f(a) < f(b) < f(c)$



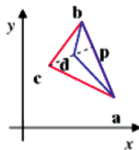
**Reflection:**  
 $d-p = -(c-p)$  with  $d-c$  perpendicular to  $b-a$ .



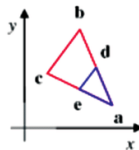
**Reflection & expansion:**  
 $d-p = -2(c-p)$  with  $d-c$  perpendicular to  $b-a$ .



**Reflection & contraction:**  
 $d-p = -\frac{1}{2}(c-p)$  with  $d-c$  perpendicular to  $b-a$ .

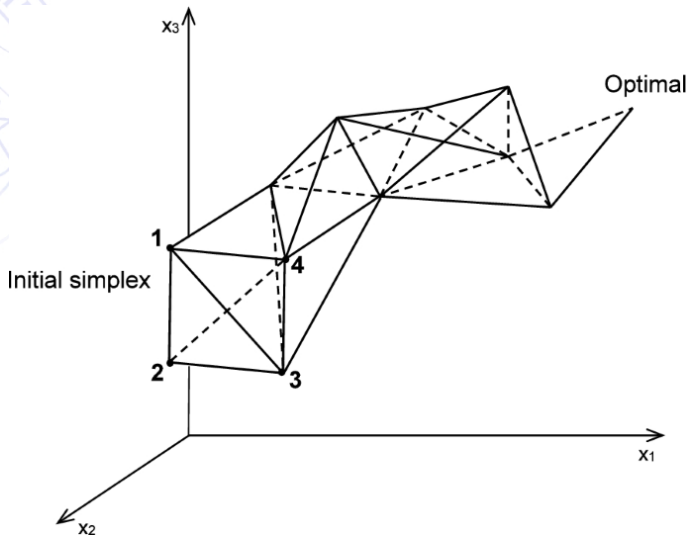


**Contraction:**  
 $d-p = \frac{1}{2}(c-p)$  with  $d-c$  perpendicular to  $b-a$ .



**Multiple contraction:**  
 $(d-a)/(b-a) = (e-a)/(c-a)$

# Simplex example



# Outline

- 
- 1 Introduction
  - 2 Bracket based methods
  - 3 Downhill Simplex
  - 4 Powell's Method
  - 5 Conjugate Descent/Gradient
  - 6 Stochastic Search
  - 7 Dynamic Programming
  - 8 Summary

# Powell's Method

- Assume you have an n-dimensional function  $f(\vec{x})$  and a starting point  $P_0$ .
- We can use the local gradient to search for an extremum
- We can generate a new estimate

$$P_{new} = P_{old} + \lambda \vec{n}$$

- Locally we can generate a Taylor expansion

$$f(x) = f(P) + \sum_i \frac{\partial f}{\partial x_i} x_i + \frac{1}{2} \sum_{ij} \frac{\partial^2 f}{\partial x_i \partial x_j} x_i x_j + \dots$$

or

$$f(x) \approx \vec{c} - b\vec{x} + \frac{1}{2} \vec{x}^T A \vec{x}$$

where

$$\begin{aligned} \vec{c} &= f(P) \\ b &= -\nabla f_P \\ A_{ij} &= \frac{\partial^2 f}{\partial x_i \partial x_j} \text{ Hessian Matrix} \end{aligned}$$

- Also remember

$$\nabla f = A\vec{x} - b$$

at an extremum

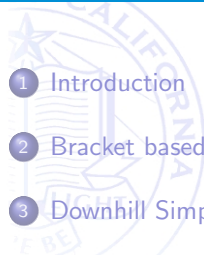
# Powell's Method

- Initialize  $N$  unit vectors

$$u_i = e_i \quad i \in 1 \dots N$$

- 1 Start at point  $P_0$
  - 2 For  $i=1$  to  $N$
  - 3 Move along  $P_i$  from  $P_{i-1}$  along  $u_i$
  - 4 New  $u_i = u_{i+1}$
  - 5 Set  $u_N = P_n - P_0$
  - 6 Move  $P_n$  to minimum value
  - 7 Make  $P_0 = P_n$
- Might generate linear degenerate solutions

# Outline

- 
- 1 Introduction
  - 2 Bracket based methods
  - 3 Downhill Simplex
  - 4 Powell's Method
  - 5 Conjugate Descent/Gradient
  - 6 Stochastic Search
  - 7 Dynamic Programming
  - 8 Summary

# Conjugate gradient descent

- If we have the gradient from

$$f(x) \approx \vec{c} - b\vec{x} + \frac{1}{2}\vec{x}^T A \vec{x}$$

- We can do a steepest descent

- 1 Start at  $P_0$
- 2 Compute  $\nabla f(P_i)$
- 3 move in the direction of gradient to point  $P_i$
- 4 repeat

- We can construct a set of conjugate vectors

$$\begin{aligned} g_{i+1} &= g_i - \lambda A h_i \\ h_{i+1} &= g_{i+1} + \gamma_i h_i \\ \lambda_i &= \frac{g_i g_i}{h_i A h_i} \\ \gamma_i &= \frac{g_{i+1} g_{i+1}}{g_i g_i} \end{aligned}$$

# Outline

- 
- 1 Introduction
  - 2 Bracket based methods
  - 3 Downhill Simplex
  - 4 Powell's Method
  - 5 Conjugate Descent/Gradient
  - 6 Stochastic Search
  - 7 Dynamic Programming
  - 8 Summary



# Stochastic Search

- So far we have used direct functional values for optimization.
- The search has been deterministic
- Sometimes the search space is too large
- What if we use a sampling based approach?
- Some possible examples
  - Traveling salesman
  - Layout of silicon for chips
- Loosely based on Boltzmann distribution

$$P(E) = \exp(-E/kT)$$

- where  $E$  is energy/entropy,  $T$  is temperature, and  $k$  is the Boltzmann constant.

# Metropolis Algorithm

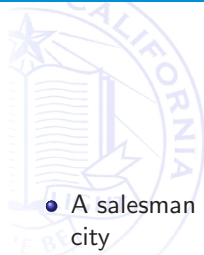
- Transformed into an algorithm by 1953 by Metropolis
- **Algorithm**
- Let  $s = s_0$
- For  $k = 0$  to  $k_{max}$ 
  - $T = \text{temperature}(k/k_{max})$
  - Pick random neighbor  $s_{new} = \text{neighbor}(T)$
  - If  $(P(S, T) \leq \text{random}(0, 1))$ 
    - $s = s_{new}$
- Return  $S$

# Simulated Annealing



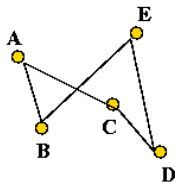
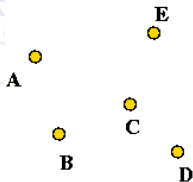
- 1 Description of possible configurations
- 2 A way to generate random perturbation of a configuration
- 3 An objective function whose minimization is the objective
- 4 A control variable that is lowered over times.

# Example - traveling salesman

- 
- A salesman has to visit  $N$  cities at locations  $(x_i, y_i)$  returning to the original city
  - Each city to be visited only once
  - Minimize the travel route
  - Problem in the optimal sense is known to be NP-hard.

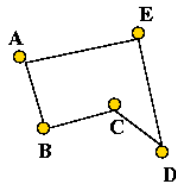
# Simple Example - Traveling Salesman

**Input:**



**A non-optimal tour:**

A B E D C



**The optimal tour:**

A B C D E

# Vacation Planning

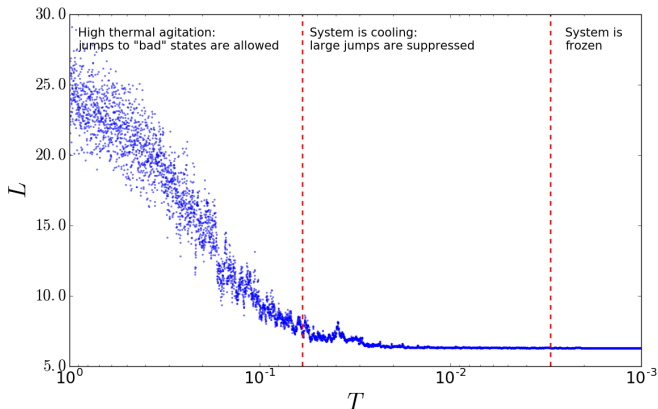
- Going to Italy for vacations



SRC [https://mirkomiorelli.github.io/SA\\_TSP/](https://mirkomiorelli.github.io/SA_TSP/)

# Vacation Planning - Temperature

- Going to Italy for vacations



SRC [https://mirkomiorelli.github.io/SA\\_TSP/](https://mirkomiorelli.github.io/SA_TSP/)

# Vacation Planning - Random initialization

- Going to Italy for vacations



SRC [https://mirkomiorelli.github.io/SA\\_TSP/](https://mirkomiorelli.github.io/SA_TSP/)



# Vacation Planning - Random initialization

- Going to Italy for vacations

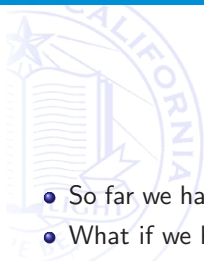


SRC [https://mirkomiorelli.github.io/SA\\_TSP/](https://mirkomiorelli.github.io/SA_TSP/)

# Outline

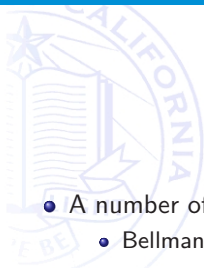
- 
- 1 Introduction
  - 2 Bracket based methods
  - 3 Downhill Simplex
  - 4 Powell's Method
  - 5 Conjugate Descent/Gradient
  - 6 Stochastic Search
  - 7 Dynamic Programming
  - 8 Summary

# Dynamic Programming



- So far we have considered functional optimization and stochastic optimization
- What if we have a limited set of action to optimize across?
- Say optimizing a set of actions to traverse a graph?
- A strategy to could be
  - Generate a cost-map across the state space
  - Backtrack to find the optimal set of actions

# Dynamic programming

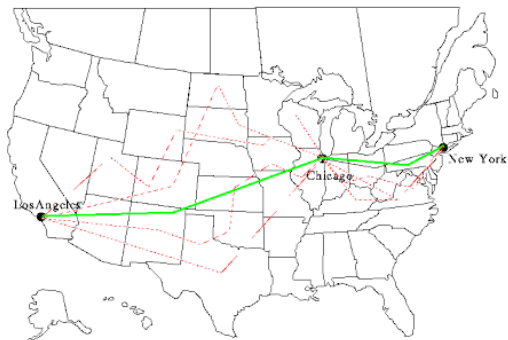


- A number of different names / approaches has been used
  - Bellman, Dijkstra, Viterbi, ...
- Selection a state space for optimization
- Identifying a set of possible actions
- Formulation of an objective function

# Example navigation

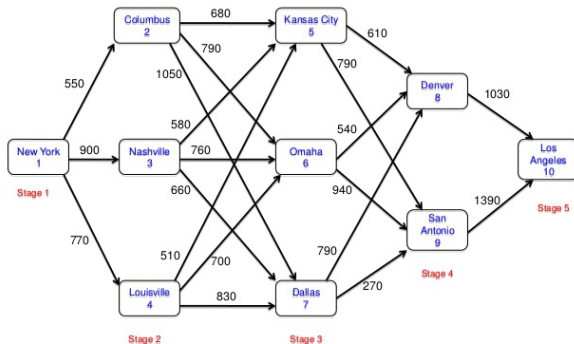


## TRIVIAL EXAMPLE OF BELLMAN'S OPTIMALITY PRINCIPLE



# Example navigation

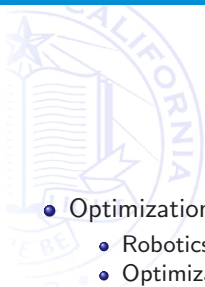
## Shortest Path: network figure



# Outline

- 
- 1 Introduction
  - 2 Bracket based methods
  - 3 Downhill Simplex
  - 4 Powell's Method
  - 5 Conjugate Descent/Gradient
  - 6 Stochastic Search
  - 7 Dynamic Programming
  - 8 Summary

# Summary



- Optimization is a key objective in robotics
  - Robotics in many cases is about formulation of a graph
  - Optimization of an objective function across the graph
- Considered deterministic and stochastic approaches to optimization
- Covered the basics and gave an impression of the fundamentals





# Questions