ROB 537:
Learning-Based Control

Week 2, Lecture 1
Search and Optimization

Announcements:
HW 1 Due TODAY
Project Background due on 10/9

Reading assignment:
EA survey paper (Flemming)

Oregon State University

Search - Optimization

- Basic premise:
  - A set of variables $\boldsymbol{x} = \{x_i, \ldots, x_N\}$ (or states)
  - A cost function C(x)
    E(x) , J(x), G(x) …
  - A set of constraints H, on the possible values of $\boldsymbol{x}$
- Find the variables $\boldsymbol{x}'$ such that:
  - $\boldsymbol{x}'$ satisfies H ; and
  - And $C(\boldsymbol{x}') > C(\boldsymbol{x})$ for all $\boldsymbol{x}$ (for maximization)

Oregon State University

Search - Optimization

- Two fundamentally different types of problems:
  - Cost is an analytic function of x
    - Example: Minimize $C(x) = 7x^4 + 3.4x^3 + 3x^2 + 4x$
    - Unconstrained: gradient descent
    - Constrained: Lagrange multipliers

Oregon State University

Search - Optimization

- Two fundamentally different types of problems:
  - Cost is an analytic function of x
    - Example: Minimize $C(x) = 7x^4 + 3.4x^3 + 3x^2 + 4x$
    - Unconstrained: gradient descent
    - Constrained: Lagrange multipliers
  - Cost is function of final state
    - Example: Traffic congestion management
      - Many vehicles
      - Roads of given capacity
      - Problem: determine departure times and path to minimize congestion

Oregon State University

Optimization is Easy

$$F(x)$$

Oregon State University

Optimization is Easy

$$F(\qquad\qquad x\qquad\qquad)$$

Oregon State University

Optimization is Easy

$$F(\quad t \quad \vec{\hat{x}} \quad)$$

Oregon State University

Optimization is Easy

$$F(\quad t \quad \vec{\hat{x}} \quad \vec{y} \quad \vec{z} \quad)$$

Oregon State University

Optimization is Easy

$$F(\quad g(t\quad \vec{\hat{x}}\quad \vec{y}\quad \vec{z})\quad)$$

Oregon State University

Optimization is Easy

$$F(\quad h(\quad \phi(g(t\quad \vec{\hat{x}}\quad \vec{y}\quad \vec{z})),\quad \theta(v)\quad)$$

Oregon State University

Optimization Maybe is not so Easy

Congestion based on air traffic controller availability

Sector congestion, based on count and path

Sector count based on path

Path based on departure time

Delay of craft entering sector

$$F(\ r(\ \ h(\ \ \phi(g(t\ \ \hat{\vec{x}}\ \ \vec{y}\ \ \vec{z}))),\ \ \theta(v,w)\ \ )$$

x: departure time vector
x^: observed departure time

Oregon State University

Search Problem: Bin Packing

- Bin Packing
  - Given object sizes, bin sizes.
  - States: contents of each bin
    - Example: ***x*** = {1.2, 0 , 2.5 , 4.1 , 3.2 , 0 , 0.5}
  - Cost function: C(***x***) = number of nonzero bins as a function of ***x***
  - Constraints: each bin must be below size
    - {1.2, 0 , 2.5 , 4.1 , 3.2 , 0 , 0.5} is not valid if bin size is 4.
  - Find object distribution ***x'*** such that: C(***x'***) < C(***x***) for all ***x***

Oregon State University

Search Problem: Traveling Salesman

- Traveling Salesman Problem:
  - You need to visit *N* cities. You want to minimize the distance traveled but still see each city once and return to the starting city.
    - Given distance matrix specifying distance between cities
  - States: order in which a city is visited
    - Example: $\boldsymbol{x}$ = {1,2,5,4,3}
  - Cost function: C($\boldsymbol{x}$) length of the trip, based on $\boldsymbol{x}$ and distance matrix
  - Constraints: each city visited exactly once
    - {1,1,3,2,1} is not a valid tour
  - Find shortest valid tour $\boldsymbol{x}'$ such that: C($\boldsymbol{x}'$) < C($\boldsymbol{x}$) for all $\boldsymbol{x}$

Oregon State University

Search - Optimization

- Local search
  - Generally applicable to any search problem
  - Use single current state
  - Improves solution at each step
  - Susceptible to local minima
- Example: Traveling Salesman Problem
  - Start with a valid tour.
  - Improve tour by swapping city orders
  - Keep going

Oregon State University

Discrete vs. Continuous Spaces

- Discrete Spaces
  - All states can be enumerated
  - 9-puzzle, chess, bin packing
- Continuous Spaces
  - Infinite number of states
  - Use algorithms that do not require enumerating states
  - Discretize neighborhood of each variable
- Most real world problems are continuous

Oregon State University

Local Search in Discrete Spaces

- Single solution-based search:
  - Hill climbing
  - Stochastic hill climbing (pick among "good" moves)
  - ε-greedy search
  - Simulated annealing
  - Tabu Search
- Population-based search:
  - Beam search
  - Stochastic beam search
  - Simplex Search
  - Particle Swarm Optimization
  - Evolutionary algorithms
  - Genetic Algorithms

Oregon State University

Hill Climbing in Discrete Spaces

- Problem Setup:
  - Objective function
  - States
- Algorithm:
  1. Start from an initial (random) state
  2. Generate (all) successor states from current state
  3. Select successor state with highest value
  4. Go to step 2
- Issues:
  - Local maxima
  - Stop/go when reaching a plateau?

Oregon State University

Stochastic Hill Climbing in Discrete Spaces

- Problem Setup:
  - Objective function
  - States
- Algorithm:
  1. Start from an initial (random) state
  2. Generate (all) successor states from current state
  3. Select a random successor state of higher value than current state
  4. Go to step 2
- Issues:
  - Slower convergence than hill-climbing
  - Potentially better solution than hill-climbing

Oregon State University

First-Choice Hill Climbing

- Problem Setup:
  - Objective function
  - States
- Algorithm:
  1. Start from an initial (random) state
  2. Generate successor state randomly from current state
  3. Select first successor state of higher value than current state
  4. Go to step 2
- Issues:
  - Implementation of stochastic hill climbing
  - Much more efficient that stochastic hill climbing when there are many successor states

Oregon State University

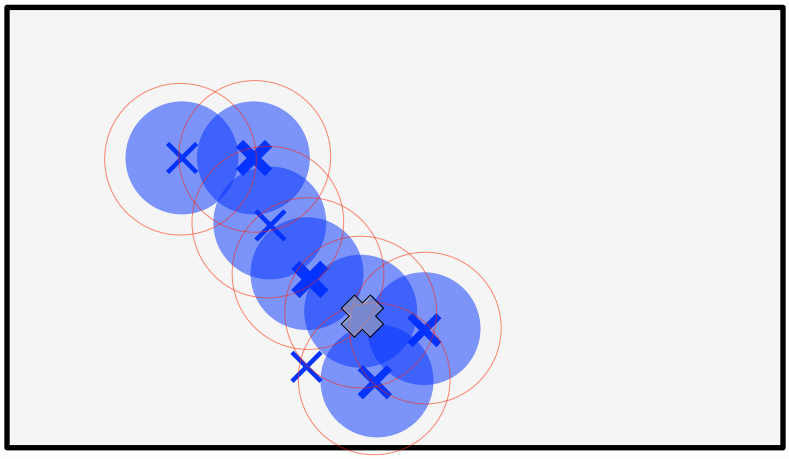
Tabu Search

- Problem Setup:
  - Objective function
  - States
- Algorithm:
  1. Start from an initial (random) state
  2. Randomly generate many candidate successor states "close" to current solution
  3. Check Tabu status of each
  4. Choose the best admissible successor state (worst solutions allowed)
  5. Update Tabu Status
  6. Goto 2 or stop if criteria is met
- Issues:
  - What if all successors are Tabu?
  - Domain knowledge intensive

Tabu Status: Solutions near recently explored solutions are not allowed

Oregon State University

Tabu Search



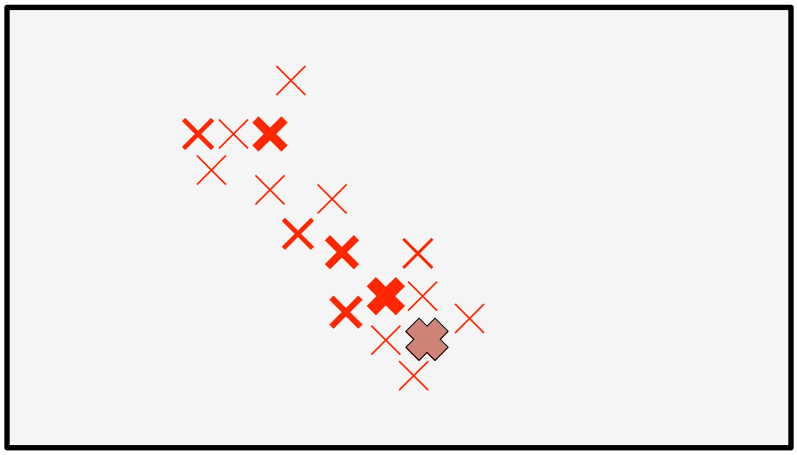
Oregon State University

Simulated Annealing

- Problem Setup:
  - Objective function
  - States
- Algorithm:
  1. Start from an initial (random) state
  2. Update Parameter T (temperature)
  3. Randomly generate a successor state from current state
  4. If successor is better than current state:
     - select it as the next state
     - Go to step 2
  5. If successor state is NOT better than current state:
     - Select it with probability p, where: $p \propto e^{-\Delta E / T}$
     - Go to step 2
- Issues:
  - Accept bad solution some of the time (based on how bad they are)
  - Accept bad solutions early
  - If T is reduced slowly enough simulated annealing finds global optimum

Oregon State University

Simulated Annealing



Oregon State University

ε-Greedy Search

- Problem Setup:
  - Objective function
  - States
- Algorithm:
  1. Start from an initial (random) state
  2. Randomly generate a successor state from current state
  3. If successor is better than current state:
     - select it as the next state
     - Go to step 2
  4. If successor state is NOT better than current state:
     - Select it with probability ε
     - Go to step 2
- Issues:
  - Accept bad solution some of the time (NOT based on how bad they are)
  - More efficient that simulated annealing (no exponential computation)
  - Can modify ε if desired

Oregon State University

Population Based Search

- All previous algorithms were based on modifying a single solution

- What if you store and search through multiple solutions at once ?

  Population based search

Oregon State University

Beam Search in Discrete Spaces

- Problem Setup:
  - Objective function
  - States
- Algorithm:
  1. Start from k initial (random) states
  2. Generate all successor states for all current states
  3. Select k best states from the pool
  4. Go to step 2
- Issues:
  - Not the same as running hill climbing k times
    - "Information" passed among k states (not directly)
  - Lack of diversity among the k states
    - States quickly concentrate on specific areas of the state space

Oregon State University

Stochastic Beam Search in Discrete Spaces

- Problem Setup:
  - Objective function
  - States
- Algorithm:
  1. Start from k initial (random) states
  2. Generate all successor states for all current states
  3. Select k states randomly from the pool, with value of state affecting its selection probability
  4. Go to step 2
- Issues:
  - Useful information passed among k states
  - Keeps more variation among k states
  - Most basic "population-based" search algorithm

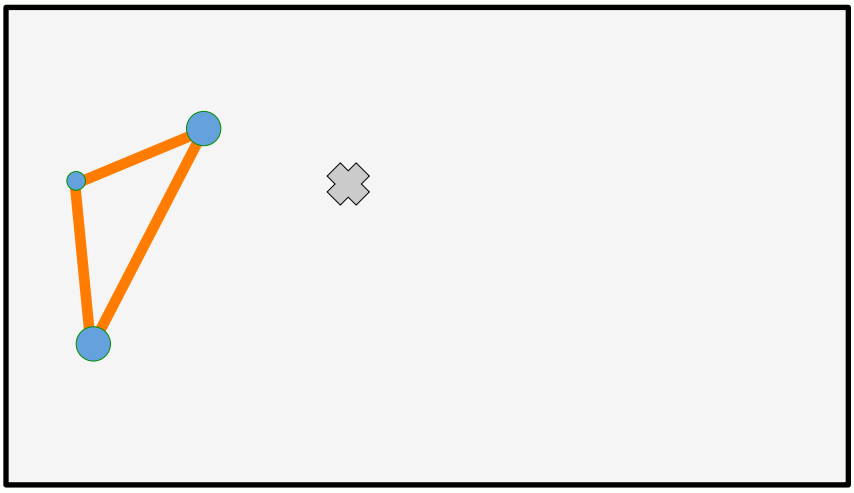
Oregon State University

Simplex Search

- Problem Setup:
  - Objective function
  - States
  - Scaling factor F=[1:2]
- Algorithm:
  1. Start from {*k = size(x)+1*} initial (random) states
  2. Evaluate all k states
  3. Discover worst solution $s^{\sim}$
  4. Calculate difference between $x(s^{\sim})$ and centroid of all other solutions $x(s^{k})$
  5. $\Delta = x(s^{\sim}) - x(s^{k})$
  6. Create $x(s') = x(s^{\sim}) - F*\Delta$
  7. Stop after desired number of iterations
- Issues:
  - Scaling in decision variables
  - Deterministic solution

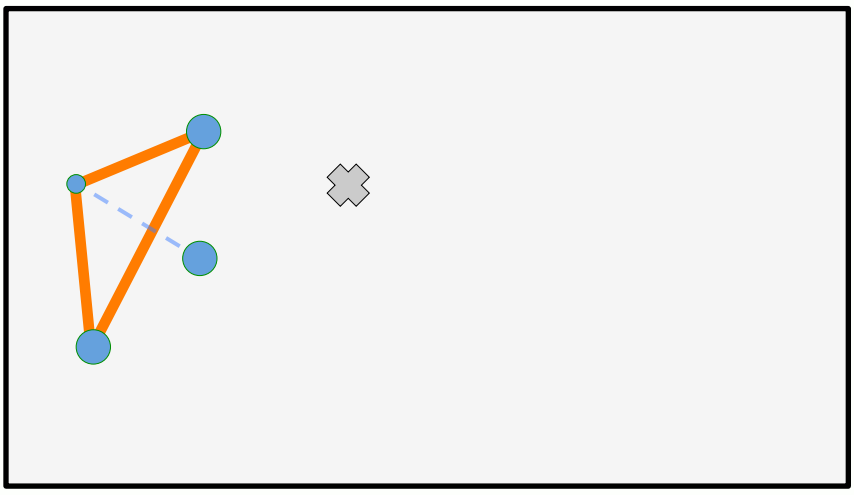
Oregon State University

Simplex Search



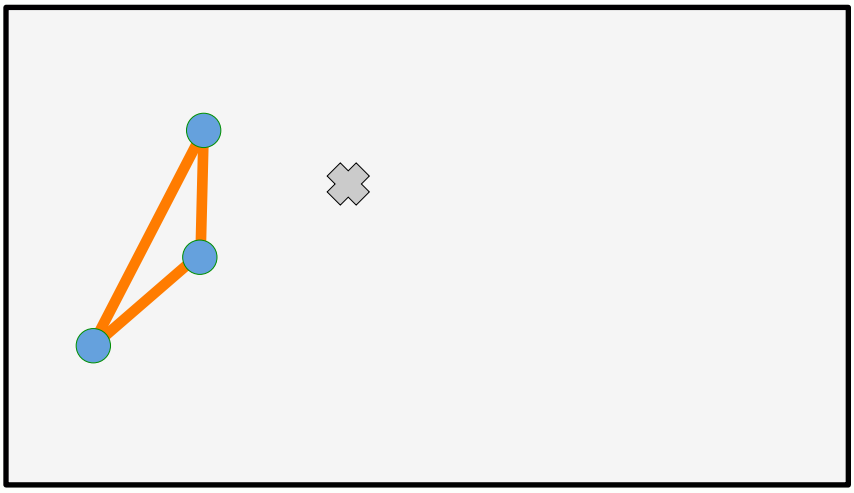
Oregon State University

Simplex Search



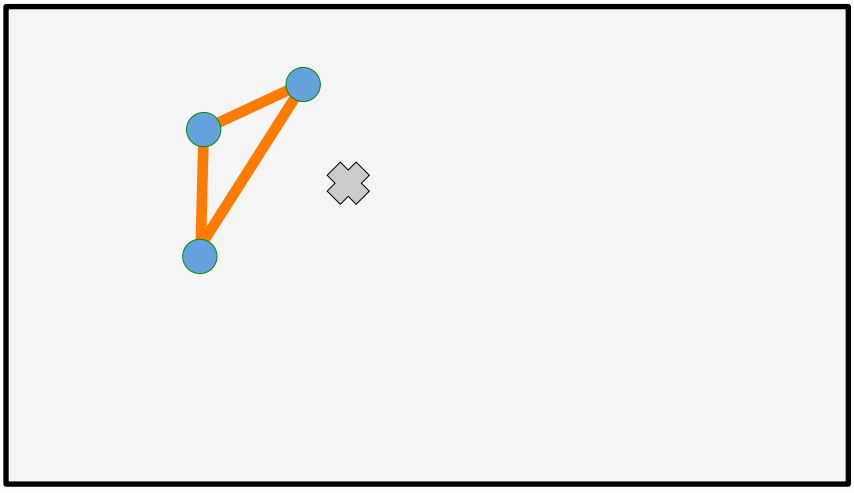
Oregon State University

Simplex Search



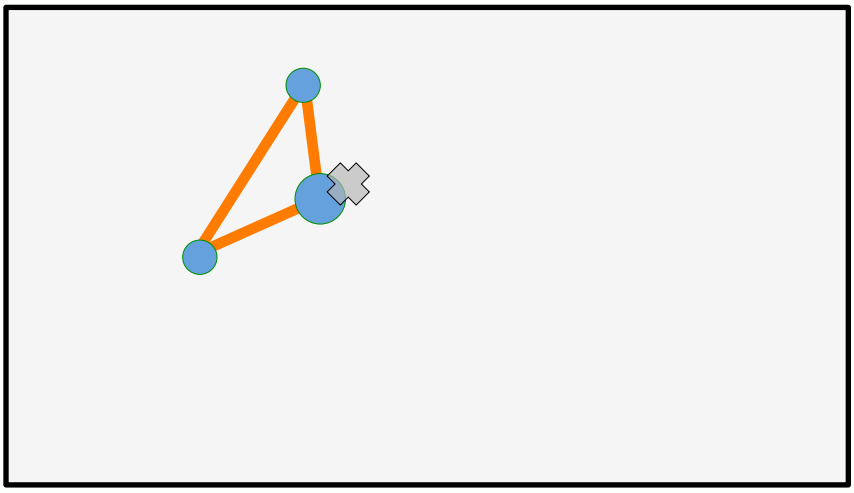
Oregon State University

Simplex Search



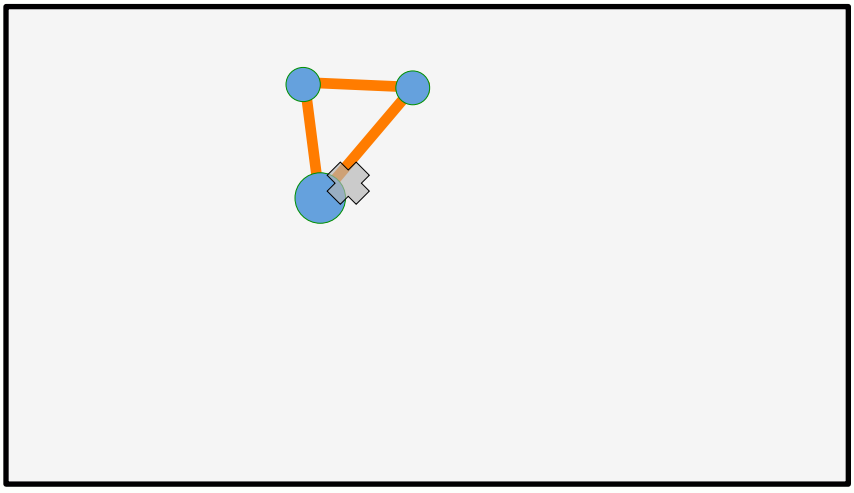
Oregon State University

Simplex Search



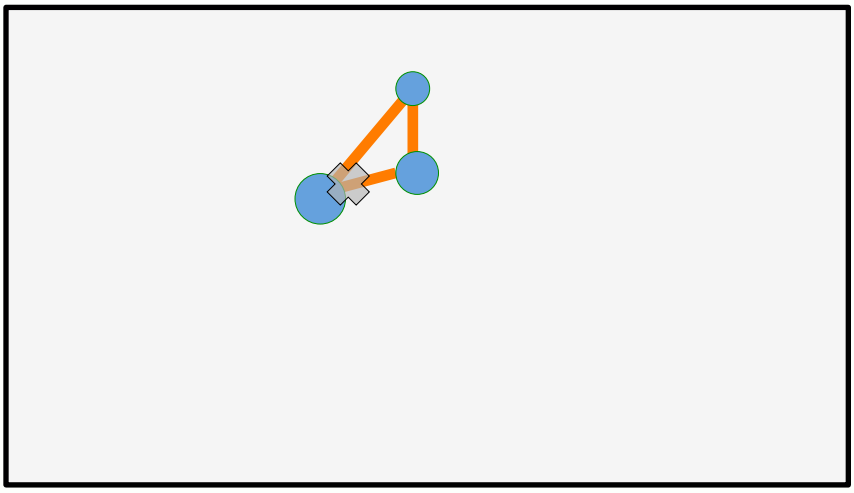
Oregon State University

Simplex Search



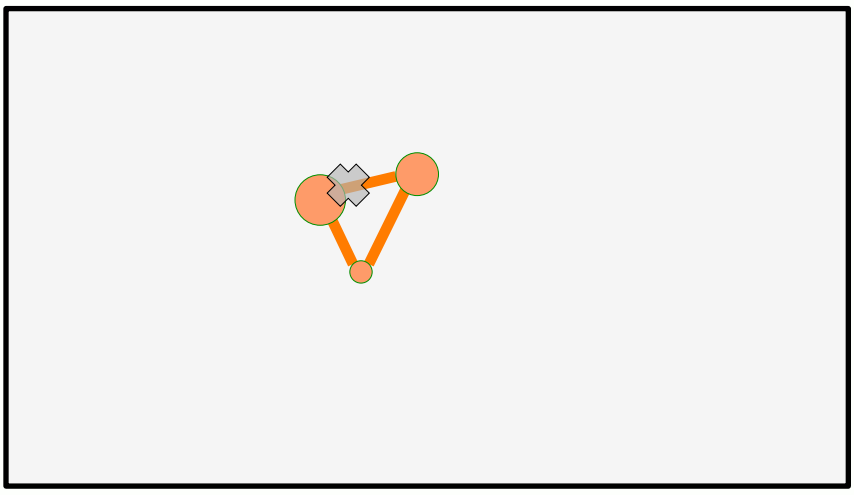
Oregon State University

Simplex Search



Oregon State University

Simplex Search



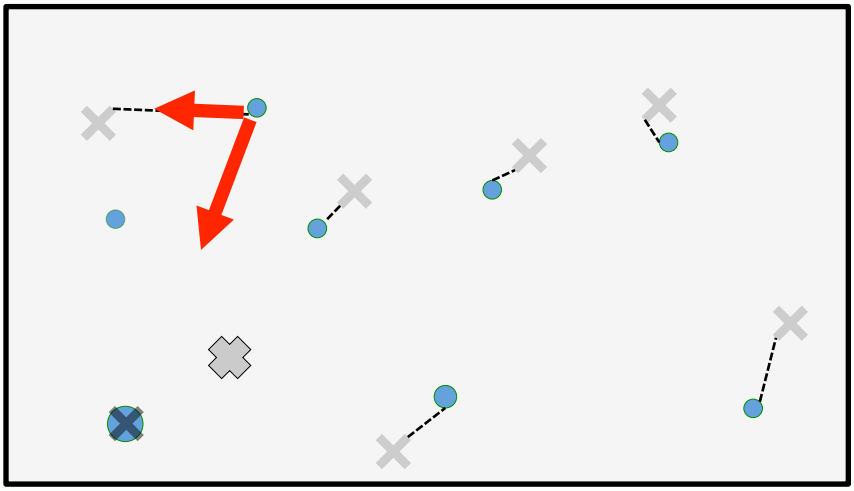
Oregon State University

Particle Swarm Optimization

- Problem Setup:
  - Objective function
  - States
- Algorithm:
  1. Start from *n* initial (random) states (particles with velocities)
  2. Choose global, personal weights ϕ_g ϕ_p
  3. For each particle
     1. Pick random numbers *p*, *g* in [0:1]
     2. Update velocity: $v = \omega v + p\,\phi_p\,(x-b_p) + g\,\phi_g(x-b_g)$
     3. Update position: $x = x + v$
     4. Track best known position b_p
  4. Track best known global position b_g
  5. Repeat steps 3,4 until convergence or stopping criteria is met
- Issues:
  - Parameters: Velocity, Momentum

Oregon State University

Particle Swarm Optimization



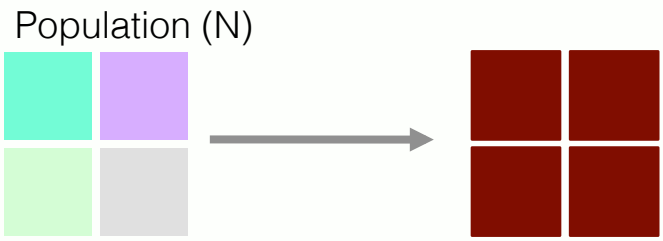
Oregon State University

Evolutionary Algorithms

- Problem Setup:
  - Objective function
  - States
- Algorithm:
  1. Start from k initial (random) states
  2. Generate k successor states randomly
  3. Perturb those states (mutation)
  4. Select k states from the pool of 2k states, with value of state affecting its selection probability
  5. Go to step 2
- Issues:
  - Addition of perturbation generates "new" solutions
  - Slower convergence than stochastic beam search
  - Potential for finding better solution due to "mutation"
  - Steps 2 and 3 are indistinguishable in most implementations

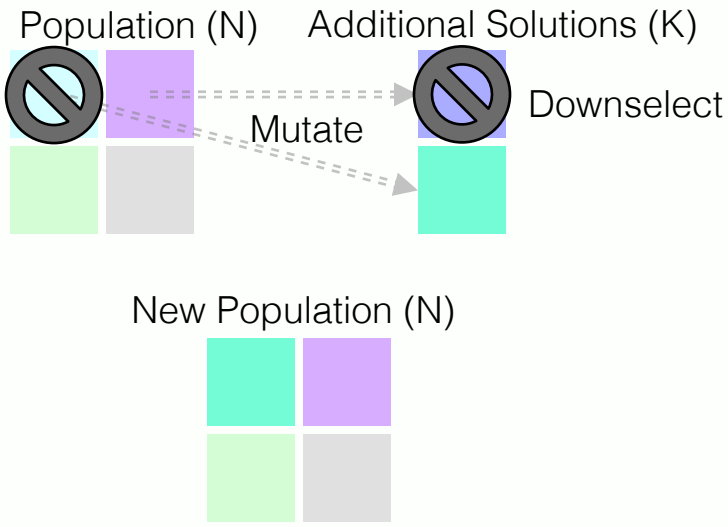
Oregon State University

Evolutionary Algorithms



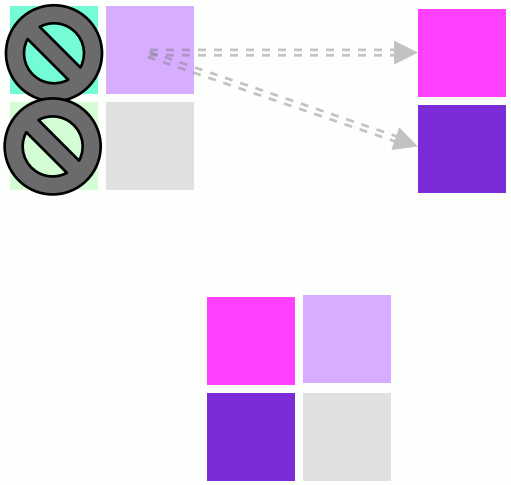
Oregon State University

Evolutionary Algorithms



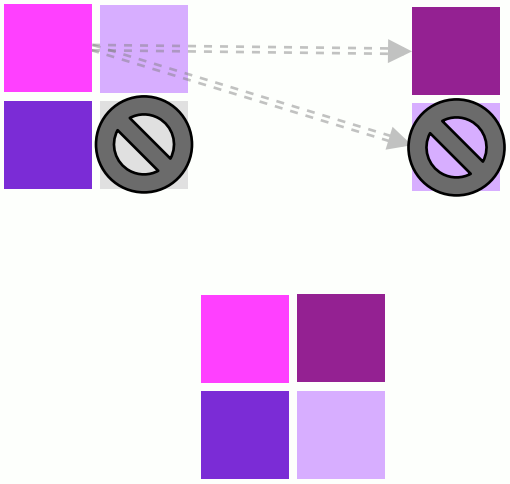
Oregon State University

Evolutionary Algorithms



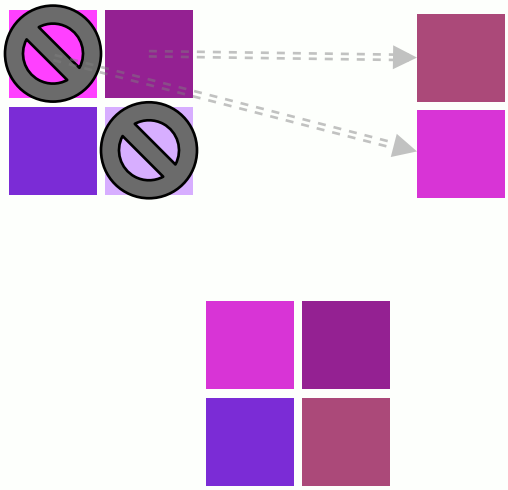
Oregon State University

Evolutionary Algorithms



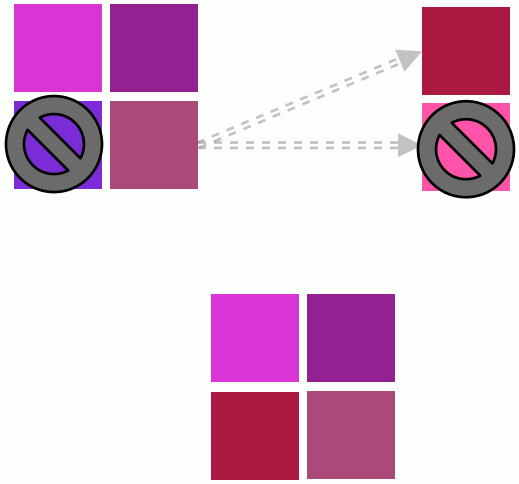
Oregon State University

Evolutionary Algorithms



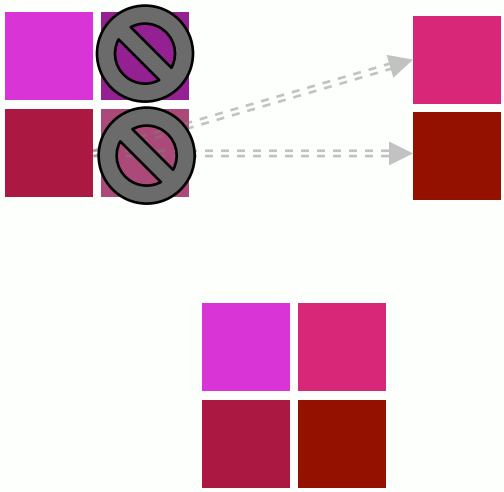
Oregon State University

Evolutionary Algorithms



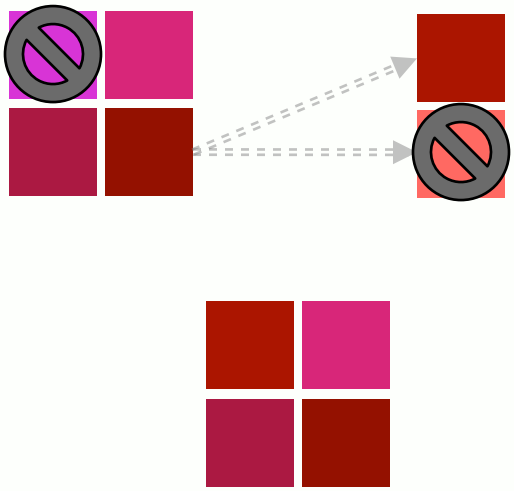
Oregon State University

Evolutionary Algorithms



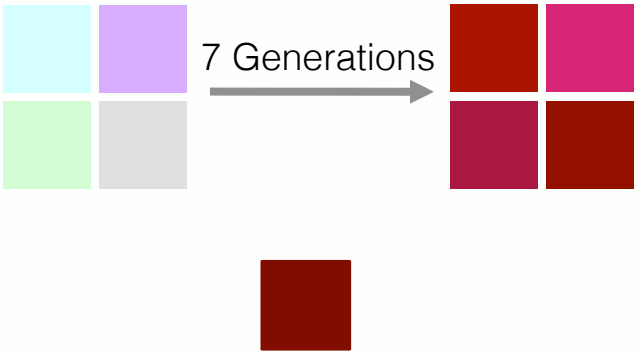
Oregon State University

Evolutionary Algorithms



Oregon State University

Evolutionary Algorithms



Oregon State University

Genetic Algorithms

- Problem Setup:
  - Objective function
  - States
- Algorithm:
  1. Start from k initial (random) states
  2. Generate k new states using current states (crossover: "merge" parts of two states to generate a new one)
  3. Perturb those states (mutation)
  4. Select k states randomly from the pool, with value of state affecting its selection probability
  5. Go to step 2
- Issues:
  - Crossover aims to preserve and combine good "partial" states
  - Addition of perturbation generates "new" solutions
  - Potential for finding better solution due to "mutation"
  - Genetic algorithms provide no "mathematical advantage"
  - Genetic algorithms may provide an implementation advantage: preserving good partial solutions allows for a "hierarchical search"

Oregon State University