Genetic Algorithms

Why we like Genetic Algorithms

- Totally generic if you do it right – All you NEED to override is the heuristic/fitness function.
  - Algorithm is separate from problem representation.

- Can find solutions to problems in very strange solution spaces.

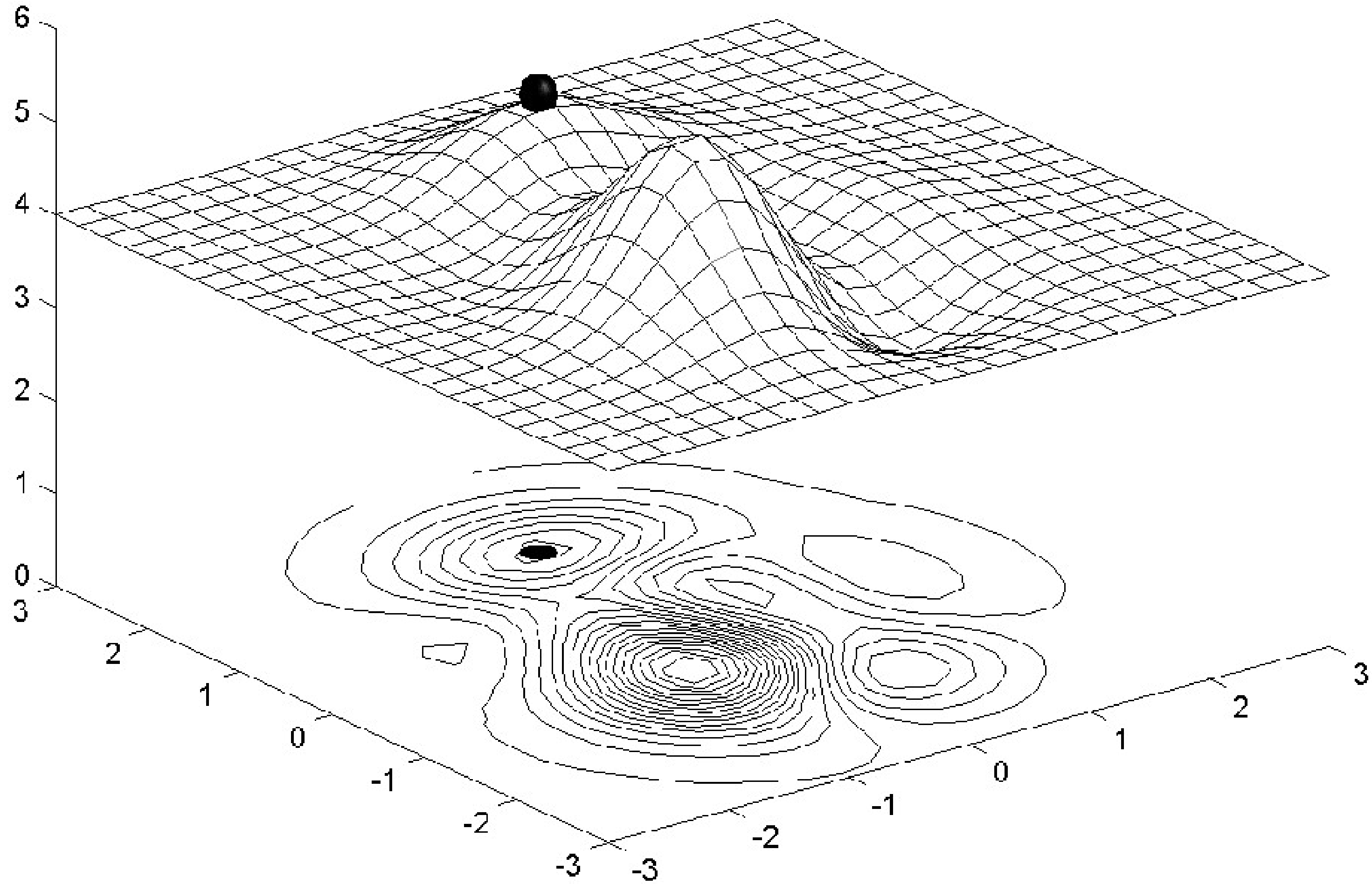
Things to Watch Out For

- Tweaks/optimizations are most likely going to be very problem specific.
- Easy to lose good members of the population

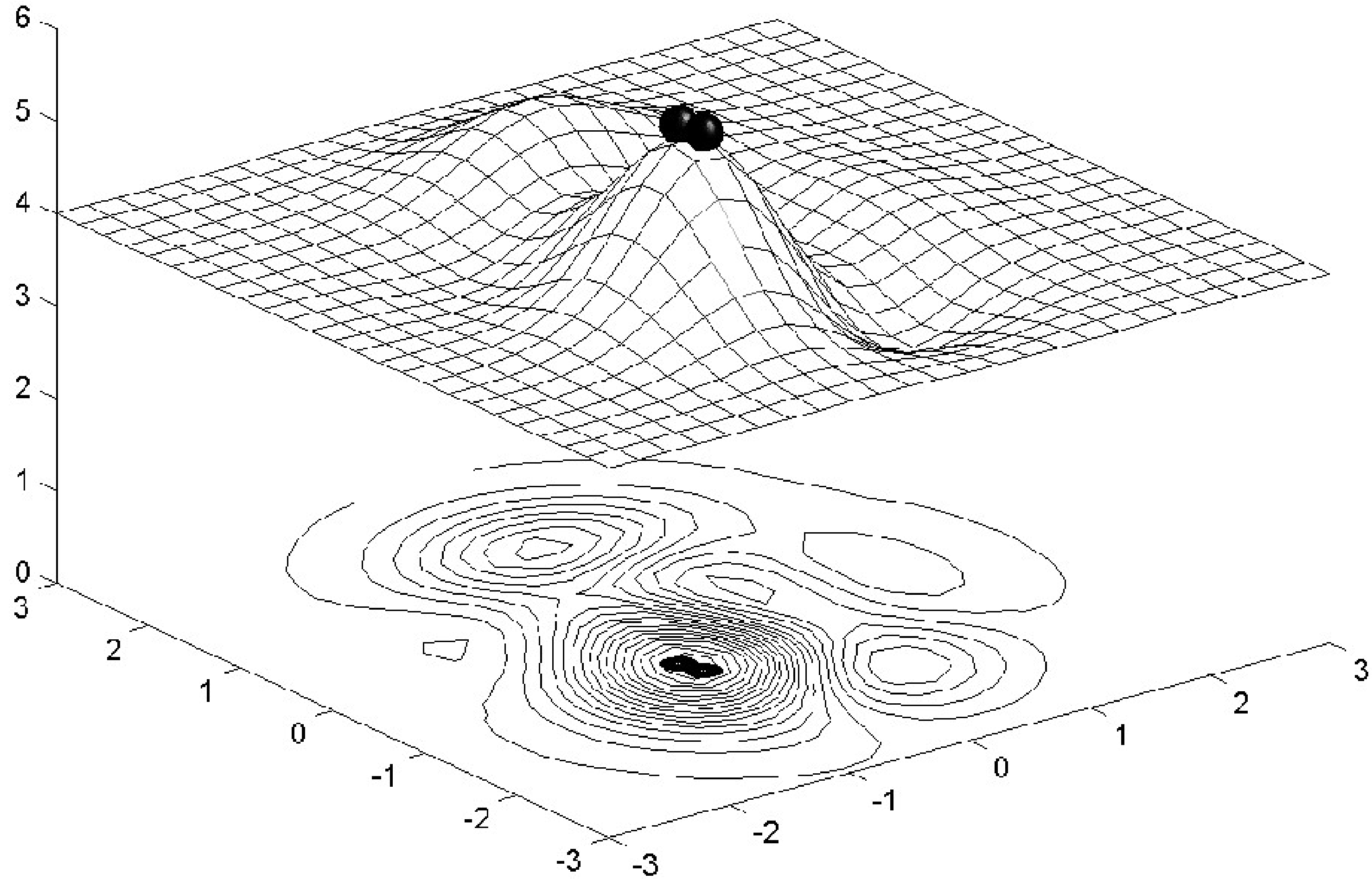
Search Capabilities

- Search Quality and Performance
  - Quality: speed to achieve solution
  - Performance: can algorithm finds solution
- Two kinds of solution GA might find:
  - Local fitness optimum: The best solution in some location of search space
  - Global fitness optimum: The best solution in the search space

Chromosome locations on the surface of the "peak" function: local maximum



Chromosome locations on the surface of the “peak” function: global maximum



Problems with fitness range

- Premature convergence
  - Fitness too large
  - Relatively superfit individuals dominate population
  - Population converges to a local maximum
  - Too much exploitation; too few exploration
- Slow finishing
  - Fitness too small
  - No selection pressure
  - After many generations, average fitness has converged, but no global maximum is found; not sufficient difference between best and average fitness
  - Too few exploitation; too much exploration

Some terms

- <u>Generation Gap</u>: The fraction of the population that is replaced each cycle. A generation gap of 1.0 means that the whole population is replaced by the offspring. A generation gap of 0.01 (given a population size of 100) means only 1 is replaced by the offspring

- <u>Selecting pressure</u>
  - Difference of fitness among individuals

Chromosome Selection Techniques

- **Elitism (***Generational* GA)
  - allow some of the better organisms from the current generation to carry over to the next unaltered
  - Otherwise, the population is filled with new children
- **Steady State Selection**
  - Insert a few new children
  - killing off some preexisting individuals to make room for them (Can cut out the weakest members)
  - Good for evolving rules
- **Fitness Proportionate Selection**
  - Roulette Wheel Selection (original technique)
- **Tournament**

Elitism

Algorithm 33 *The Genetic Algorithm with Elitism*

```
popsize ← desired population size
n ← desired number of elite individuals          ▷ popsize − n should be even

P ← {}
for popsize times do
    P ← P ∪ {new random individual}
Best ← □
repeat
    for each individual P_i ∈ P do
        AssessFitness(P_i)
        if Best = □ or Fitness(P_i) > Fitness(Best) then
            Best ← P_i
    Q ← {the n fittest individuals in P, breaking ties at random}
    for (popsize − n)/2 times do
        Parent P_a ← SelectWithReplacement(P)
        Parent P_b ← SelectWithReplacement(P)
        Children C_a, C_b ← Crossover(Copy(P_a), Copy(P_b))
        Q ← Q ∪ {Mutate(C_a), Mutate(C_b)}
    P ← Q
until Best is the ideal solution or we have run out of time
return Best
```

Steady State Selection

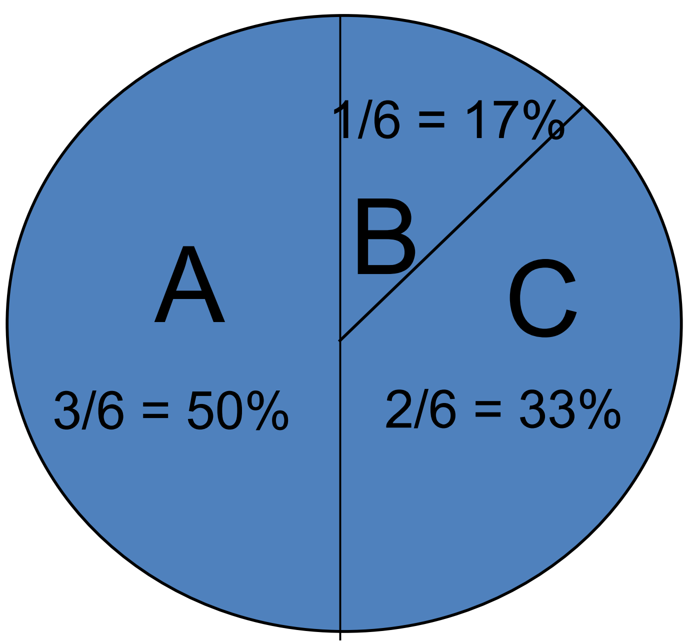
Algorithm 34 *The Steady-State Genetic Algorithm*

$popsize \leftarrow$ desired population size

$P \leftarrow \{\}$
for $popsize$ times **do**
    $P \leftarrow P \cup \{$new random individual$\}$
$Best \leftarrow \square$
for each individual $P_i \in P$ **do**
    AssessFitness(P_i)
    if $Best = \square$ or Fitness(P_i) $>$ Fitness($Best$) **then**
        $Best \leftarrow P_i$
repeat
    Parent $P_a \leftarrow$ SelectWithReplacement(P)   ▷ We first breed two children C_a and C_b
    Parent $P_b \leftarrow$ SelectWithReplacement(P)
    Children $C_a, C_b \leftarrow$ Crossover(Copy(P_a), Copy(P_b))
    $C_a \leftarrow$ Mutate(C_a)
    $C_b \leftarrow$ Mutate(C_b)
    AssessFitness(C_a)   ▷ We next assess the fitness of C_a and C_b
    if Fitness(C_a) $>$ Fitness($Best$) **then**
        $Best \leftarrow C_a$
    AssessFitness(C_b)
    if Fitness(C_b) $>$ Fitness($Best$) **then**
        $Best \leftarrow C_b$
    Individual $P_d \leftarrow$ SelectForDeath(P)
    Individual $P_e \leftarrow$ SelectForDeath(P)   ▷ P_d must be $\neq P_e$
    $P \leftarrow P - \{P_d, P_e\}$ ▷ We then delete P_d and P_e from the population
    $P \leftarrow P \cup \{C_a, C_b\}$ ▷ Finally we add C_a and C_b to the population
until $Best$ is the ideal solution or we have run out of time
return $Best$

Roulette Wheel Selection

- Main idea: better individuals get higher chance
  - Chances proportional to fitness
  - Implementation: roulette wheel technique
    - » Assign to each individual a part of the roulette wheel
    - » Spin the wheel n times to select n individuals



fitness(A) = 3

fitness(B) = 1

fitness(C) = 2

Scaling Techniques

- Instead of using the raw fitness score, run it through a function first.
  - Rank Scaling (Linear Rank Scaling)
  - Sigma Scaling

Rank Scaling (Linear Scaling)

- Order results by fitness, rescore based on rank.

  New score $= (P-r_i)(max-min)/(P-1) + min$
  - Where r_i is the rank of individual i,
  - P is the population size,
  - Max represents the fitness to assign to the best individual,
  - Min represents the fitness to assign to the worst individual.
- Better selecting pressure while population converges
  - Tend to avoid premature convergence by tempering selection pressure for large fitness differentials that occur in early generations By amplifying small fitness differences in later generations, selection pressure is increased compared to alternative selection strategies.
- Worse distinguishability for individuals with higher difference of fitness. Thus, convergence might be slow and search might be directed to wrong direction
- Another disadvantage associated with linear rank selection is that the population must be sorted on each cycle.

Sigma Scaling

- **Sigma Scaling**
  - This method keeps the selection pressure relatively constant
    - it is not too strong in early generations and not too weak once the population has stabilized and fitness differences are smaller
  - An individual's expected value is a function of its fitness, the population mean, and the population standard deviation.

$$ExpVal(i,t) = \begin{cases} 1 + \dfrac{f(i) - \bar{f}(t)}{2\delta(t)} & if\, \delta(t) \neq 0 \\ 1.0 & if\, \delta(t) = 0 \end{cases}$$

Tournament Selection

- All other methods rely on global population statistics
  - Could be a bottleneck esp. on parallel machines
  - Relies on presence of external fitness function which might not exist: e.g. evolving game players
- Procedure for Tournament selection:
  - Pick *k* members at random then select the "best" of these
  - Repeat to select more individuals

Tournament

- Binary tournament
  - Two individuals are randomly chosen; the fitter of the two is selected as a parent
- Probabilistic binary tournament
  - Two individuals are randomly chosen; with a chance p, $0.5<p<1$, the fitter of the two is selected as a parent
- Larger tournaments
  - n individuals are randomly chosen; the fittest one is selected as a parent

- By changing n and/or p, the GA can be adjusted dynamically

Tournament selection