Genetic Algorithms for Multidimensional $S$

- Let the decision be multidimensional so $S=(S_1, \ldots, S_n)$ is a vector. How do we represent this as a binary string?
- Assume $S_i$ is an integer and $0 \leq S_i < 2^k - 1$, then you can represent each $S_i$ by a binary string of length $\underline{K}$?
- Assume the length is 4 for each of three variables, then your binary string is something like: $S = [\underline{0110} | \underline{1111} | \underline{0001} | \ldots]$ where the substrings are divided by a line.
- And each of the variables is $\underline{15}$ if $k=4$.

\[ S_1 \quad S_2 \quad S_3 \]
Warehouse Example on Board

• Assume you want to consider building a warehouse at each of Z locations (i=1, ...,Z).

• The size of the warehouse can be between 10,000 sq ft up to 150,000 sq ft so $S_i = j$ means the warehouse size is $j \times 10,000$ sq ft.

• However, I need to convert 1 to 15 into a binary string to solve this integer problem with Genetic Algorithms. $\Rightarrow k = 4$ 

• How long is the string for each $S_i$? $Z \times k$
Warehouse Problem

• Assume I want to decide about building a warehouse in 4 locations. So if

• warehouse  j(base 10) j(base 2)

• A  5 0101
• B  3 0011
• C  12 1100

Then S=(0101|0011|1100), which is a binary string of length 12.
So How do you Do Crossover with Multidimensional S?

If $S = (s_1, s_2, s_3)$

- Then you do all crossover operations for the part of the string related to one decision $S_i$ (e.g. between the blue lines).
- So if crossover is at position 2 for each variable, a child could be

```
  1101 | 0001 | 1011
```

Parent 1
Parent 2
Elitism in Genetic Algorithms (used both for single and multiple dimensional problems)

- Suppose you find a solution $S_i$ that is very good (has a low fitness).
- In the algorithm we have described the old population is discarded. Is that a good idea?
- We can modify this. How should we do it?

> pick the elite of the current generation to accompany the children. This also means to control population size, we have to drop the worst children.
How large should E be?  
*E is number of “elite” parents brought to next generation*

• Advantages of large E?  
  ⇒ *preserve good solutions*  

• Disadvantages of large E?  
  ⇒ *limits diversity as there aren’t many children*  
  ⇒ *search of search space negatively impacted*  

• What do people usually use for E?  
  ⇒ *Idea: dynamically change E with time*  
  ⇒ *Stable configurations usually have E < 50%*
Review of GA-Steps in the Algorithm

• 0. Initialization. Pick algorithm parameters and initial population
• Compute fitness $F(\text{CurSk})$ for all members of current population $\{\text{CurSk} | k=1,M\}$.
• Crossover:
  – Use fitness in random selection of parents. So probability of CurSk being selected as a parent is higher if $F(\text{CurSk})$ is high for both roulette and tournament selection.
  – Use crossover to create “children” (also called offspring) from each pair of parents.
• Use mutation to make some random change in the children.
• All children become the current population and go to step 1. (The previous population is now discarded.)
• Can be modified for elitism.
Pseudo Code for Genetic algorithm
(page 120 of xeroxed reading from S&Y Text)

Procedure (Genetic Algorithm)

M = Population size.
N_g = Number of generations.
N_o = Number of offsprings.
P_μ = Mutation probability.
Ψ ← Ξ(M)

For j = 1 to M
   Evaluate f(Ψ[j])
EndFor

For i = 1 to N_g
   For j = 1 to N_o
      (x, y) ← Φ(Ψ)
      offspring[j] ← χ(x, y)
      Evaluate f(offspring[j])
   EndFor
   For j = 1 to N_o
      mutated[j] ← μ(y)
      Evaluate f(mutated[j])
   EndFor
   \( \mathcal{P} \leftarrow \text{Select}(\mathcal{P}, \text{offsprings}) \)
   \( \mathcal{P} \}
EndFor
Return highest scoring configuration in \( \mathcal{P} \).

Figure 3.3: Structure of a simple genetic algorithm.
Problems with Neighborhoods of Binary Strings used to represent integers greater than 1

• Recall that a binary string can represent an integer in base 10.

• Assume CurS = 1100 = 1\times2^3 + 1\times2^2+0\times2^1+0\times2^0 = 10

• Neighborhood from flipping one digit: 0100, 1000, 1110, 1101

<table>
<thead>
<tr>
<th>(\Delta = 6)</th>
<th>(\Delta = 2)</th>
<th>(\Delta = 1)</th>
<th>(\Delta = 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>8</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

What can you say about this range?

\(\Rightarrow\) neighborhood may be very asymmetric
\(\Rightarrow\) is this okay? \(\Rightarrow\) problem specific
Both Binary Coded GA and Tabu Search Use Binary Strings

- For some problems the variables are really binary (0 or 1).
- However, in other cases the variables are real valued or integer or something else (e.g. perturbation). In these cases the neighborhoods associated with the binary string representation does give an uneven perturbation. It might not cause a problem, but you should think about whether it might cause a problem.
- For simulated annealing there is no need to represent S as a binary string.
Questions

- Review: What are the parameters for binary coded GA?
- So what happens to the mutations? What if it is a “bad” mutation?
- What happens to children of crossover operations that are not good?
- What are some of the differences between GA and SA?
Selecting Parameters for GA and Real-Coded GA

- Setting parameters for GA is less straightforward than for SA.
- We will later discuss some methods that can be used for this.
- We have discussed here only the application of GA to integers or binary strings. Later we will discuss other applications including “Real-Coded” GA for decision variables that are real numbers.
We’ll Return Later to Genetic Algorithms

• We will return later to details of GA including setting parameter values and representation of decision variables that are not binary strings.

• Next I want to start on the third major heuristic algorithm Tabu Search so we have covered the three most widely used algorithms: SA, GA, and TS so you can start on your projects.
Project

• Teams of 2 to 4 students (unless PhD student with own project)
• Computer module provided for your “application”
• You will compare GA, SA, TS and your own improvements in these algorithms
• Later lectures will discuss how to do comparisons
• Written Report
• Short Oral Presentation
Project Topics

– **Satisfiability Testing for Artificial Intelligence**: Prof. Bart Selman, Computer Science

– **Resource Allocation in Cellular Networks**: Prof. Steve Wicker, Electrical Engineering

– **Optimal Control of Finite Element Systems with Environmental Applications**: Prof. Christine Shoemaker, Civil and Environmental Engineering

– **Protein Folding**: Molecular and Cell Biology

– **ORIE Project** (maybe Job Shop Scheduling)
General Project Instructions

• By Thursday we will have posted the general instructions for the projects, which includes all the deadlines.

• Be sure to attend Prof. Wicker’s lecture this Wednesday on heuristic optimization for optimizing cell phone networks.

• This is one possible topic for a projec.
Homeworks, Exams, Guest Lectures

• There will be homework (and possibly exam) questions on the two guest lectures (Sept. 14 and 21) so everyone should hear these lectures (not just the people doing those projects).

• The reason is that part of the process of learning heuristic methods is see by example how you convert a real problem into something that can be analyzed by optimization.