Planning and Search

Genetic algorithms

Outline

- ♢ Genetic algorithms
- ♢ Representing states (individuals, or chromosomes)
- ♢ Genetic operations (mutation, crossover)
- ♢ Example

Stochastic local beam search recap

Idea: keep k states instead of 1

choose k of all their successors, stochastically favouring better ones

Observe the close analogy to natural selection!

Genetic algorithms

= stochastic local beam search + generate successors from **pairs** of states

k states (individuals) - population

original population randomly generated

each individual represented as a string (chromosome)

each individual rated by an objective function (fitness function)

probability of being chosen for reproduction directly proportional to fitness

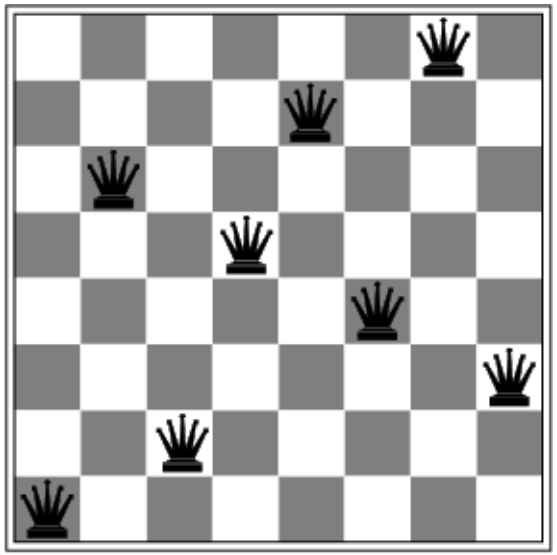
two parents produce offspring by crossover

then with some small probablity, mutation (bits of the string changed)

Example: 8 queens problem states

States: assume each queen has its own column, represent a state by listing a row where the queen is in each column (digits 1 to 8)

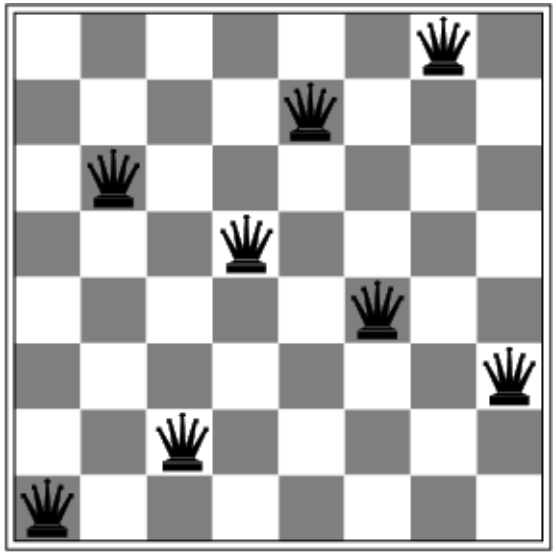
for example, the state below will be represented as 16257483



Example: 8 queens problem fitness

Fitness function: instead of $-h$ as before, use the number of **nonattacking** pairs of queens. There are 28 pairs of different queens, smaller column first, all together, so solutions have fitness 28. (Basically, fitness function is $28 - h$.)

for example, fitness of the state below is 27 (queens in columns 4 and 7 attack each other)



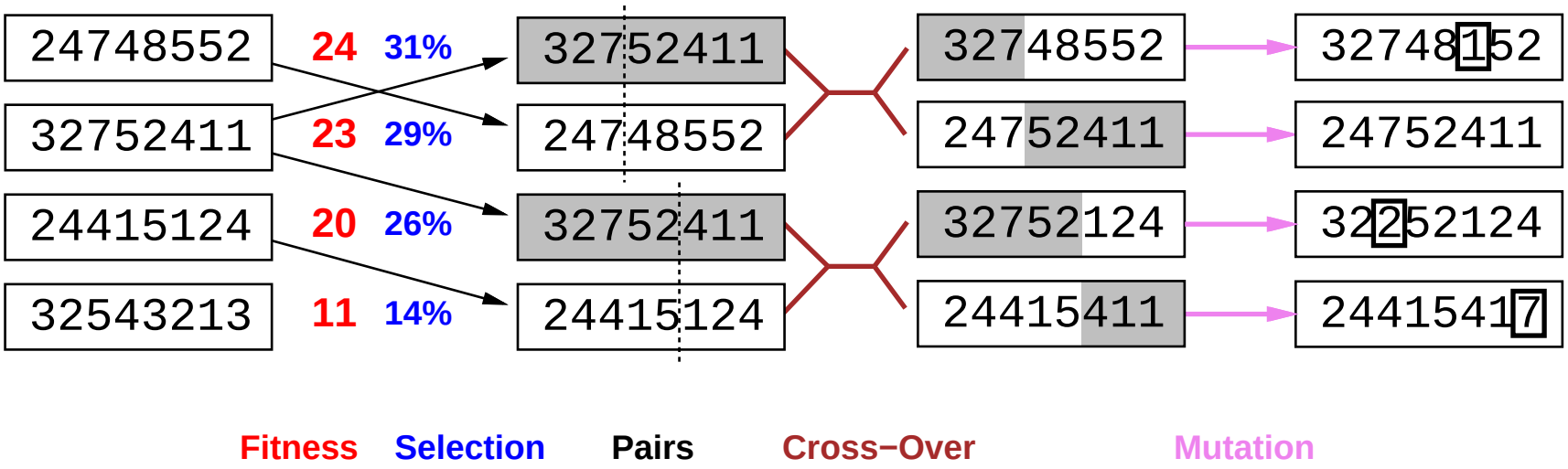
Example: 8 queens problem crossover

choose pairs for reproduction (so that those with higher fitness are more likely to be chosen, perhaps multiple times)

for each pair, choose a random crossover point between 1 and 8, say 3

produce offspring by taking substring 1-3 from the first parent and 4-8 from the second (and vice versa)

apply mutation (with small probablity) to the offspring

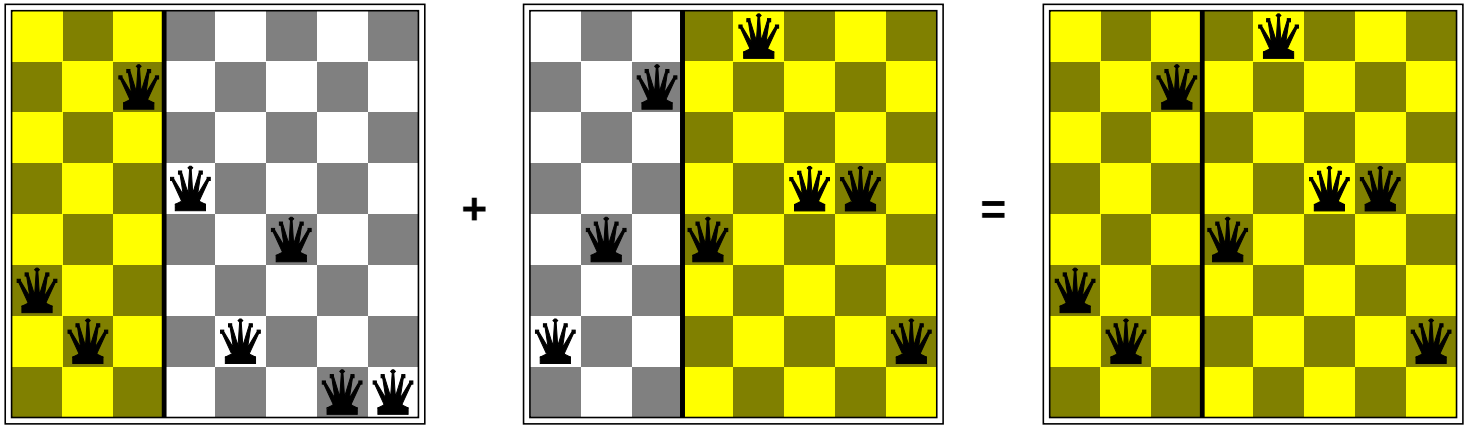


Importance of representation

Parts we swap in crossover should result in a well-formed solution (and in addition better be **meaningful**)

consider what would happen with binary representation (where position requires 3 digits)

also, chosen representation reduced search space considerably (compared to representing each square for example)



Genetic algorithm pseudocode

(this is one offspring per pair version, unlike in the example)

```
function GENETIC-ALGORITHM(population,FITNESS-FN) returns an individ-
ual
    inputs: population, a set of individuals
            FITNESS-FN, a function that measures the fitness of an individual
    repeat
        new-population ← empty set
        for i=1 to SIZE(population) do
            x ← RANDOM-SELECTION(population,FITNESS-FN)
            y ← RANDOM-SELECTION(population,FITNESS-FN)
            child ← REPRODUCE(x, y)
            if (small random probability) then child ← MUTATE(child)
            add child to new-population
    until some individual is fit enough or enough time has elapsed
    return the best individual in population, according to FITNESS-FN
```

Reproduce pseudocode

```
function REPRODUCE( x, y) returns an individual
    inputs: x, y, parent individuals
    n ← LENGTH(x); c ← random number from 1 to n
    return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, c+1, n))
```

Schemata

Schema is a partial specification of a substring

for example, 246***** is a schema describing states where the first 3 queens are in rows 2, 4 and 6

states which match the schema are called instances of the schema

genetic algorithms work best where schemata correspond to meaningful components of a solution (and crossover combines substrings corresponding to schemata)

example: combine wheels from this car with the engine of that car (not one wheel from one and three wheels from another)

Genetic algorithms summary

GAs require states encoded as strings

Representation and choice of operations and other parameters (probabilities) matters a lot

Crossover is only useful if the swapped substrings a large chunks which evolved for some purpose and are meaningful

Some precise charaterisations (of which parameters better for what kind of problems) but a lot of GA is still an art

Self-test question

Give the name of the algorithm which results from:

genetic algorithm with population size N=1.

Example

How would you solve the following problem using a genetic algorithm?

Evolve an arithmetical expression (using digits 0-9, +,-, * and /) which evaluates to number 42. Use a binary encoding. What would be a suitable fitness function?

(Problem and solutions are by Mat Buckland http://www.ai-junkie.com)

Next lecture

Search and SAT (chapter 7, mostly 7.4 and 7.6, in the 3rd edition.)