CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

Genetic Algorithm

Alexandre Bergel
DCC - University of Chile
http://bergel.eu

26/10/2020

Some benefits of Genetic Algorithm

GA provides a compelling way to *not be trapped in local optima*

GA allows optimization of systems in which *variables may be discrete or categorical*, and not only continuous

e.g., direction of a robot or characterization of an antenna segment

GA can be combined with *other AI techniques*

Example: Using GA to convert a number to binary

E.g., 8 = 0b1000, 12 = 0b1100

How to use GA to convert decimal values in a binary format?

Example: Using GA to convert a number to binary

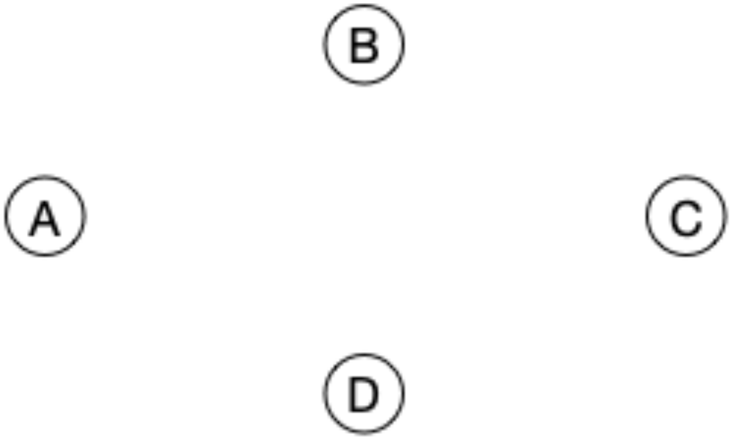
```python
NUMBER_TO_CONVERT = 121
NUMBER_OF_GENES = 10

def fitness_bits(anIndividual):
    result = 0
    exp = 1
    for v in anIndividual[::-1]:
        result += int(v) * exp
        exp *= 2
    return - abs(NUMBER_TO_CONVERT - result)

def gene_factory():
    if(random.random() > 0.5):
        return '1'
    else:
        return '0'

def sequence_bit_factory():
    return [ gene_factory() for i in range(NUMBER_OF_GENES)]

ga = GA(pop_size=100,mutation_rate=0.1,fitness=fitness_bits,
    individual_factory=sequence_bit_factory, gene_factory=gene_factory,
    termination_condition = lambda f : f == 0, silent = False, max_iter=10)

best_fitness_list, avg_list, best_individual = ga.run()
print(''.join(best_individual))
```

Example: Traveling Salesman Problem

The *Traveling Salesman Problem* is a classical algorithm problem.

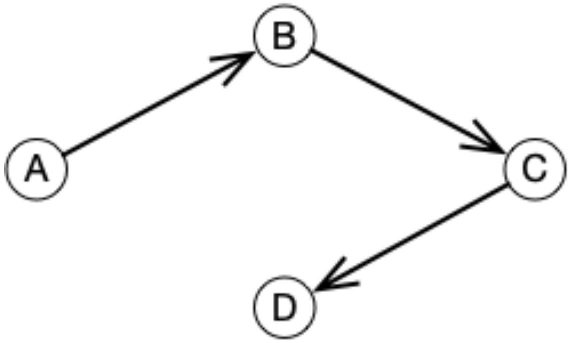
It consists in *visiting a number of cities* using the *shortest possible route*.

Example: Traveling Salesman Problem

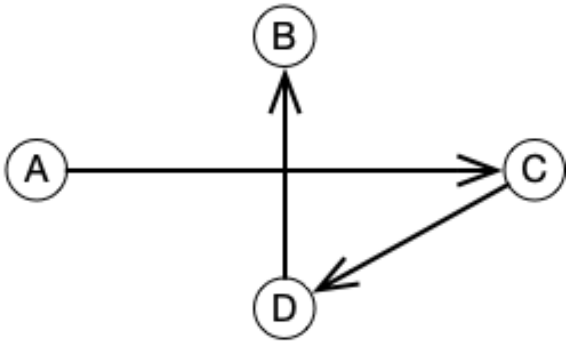


Example: Traveling Salesman Problem

ABCD

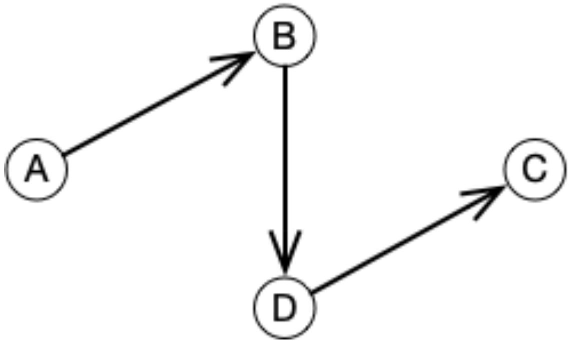


ACDB

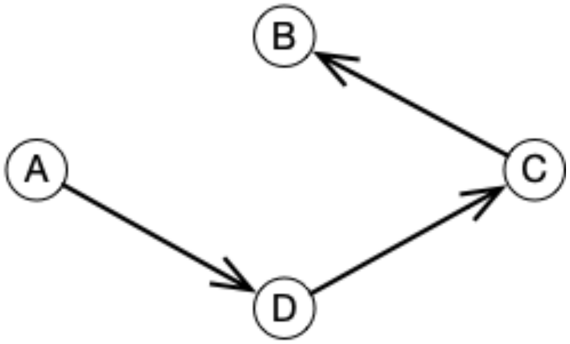


Some examples assuming we start in city A

ABDC



ADCB



Example: Traveling Salesman Problem

The Traveling Salesman Problem was *formulated in 1930*

One of the most studied algorithms

Many applications of this problem (resources planning, DNA sequencing, microchip manufacturing)

Example: Traveling Salesman Problem

TSP is apparently simple (*simple rules*)

TSP is however very difficult to solve (*need to try all combinations*)

TSP is considered as *NP-Hard*

NP-Hard means:

- Two solutions are easy to verify
- There is no-efficient way to solve the problem

Example: Traveling Salesman Problem

Finding a solution for a few cities is easy

However, *for any number of cities we do not know how to do it*

The current world record is 3038 cities

Used 50 workstations

Required 1.5 years of computation

Example: Traveling Salesman Problem

If someone, one day, solves the TSP, this would profoundly impact the World we live in

Example: Traveling Salesman Problem

How would *you* solve the TSP using genetic algorithm?

Example: Traveling Salesman Problem

How to *encode* the TSP problem?

We need to *force the algorithm to visit all cities, only once*

Our encoding *should enforce this rule*

If not, then the algorithm has to do the job itself, which is not trivial

We should use *GA to explore valid paths, not verifying whether a path is valid or not*

Example: Traveling Salesman Problem

We need two new genetic operations

Swap Mutation Operation: swap two genes in an individual. E.g., if we have ABCD, then BACD and CBAD are valid mutations. But AACD is not a valid mutation

Ordered Crossover Operation: no duplication can occur. E.g., if we have two individuals i_1 = ABCDE and i_2 = AEDBC. We pick two indices in i_1, copy the portion in the result (e.g., ..CD.), and take the genes from i_2 that are not in the new individual yet, in the same order (e.g., AECDB)

Example: Traveling Salesman Problem

These two operations ensure that *every individual is a valid path*

The algorithm can therefore be used to *optimize valid paths*

Concluding words

To solve complex problem using Genetic Algorithms, it is important to:

- define an adequate encoding
- consider new genetic operations, if necessary

dcc

CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

www.dcc.uchile.cl

/ DCCUCHILE