Evolutionary algorithms

Orsolya Sáfár

Scheme theorem, Gray coding, Hypothesis of building blocks

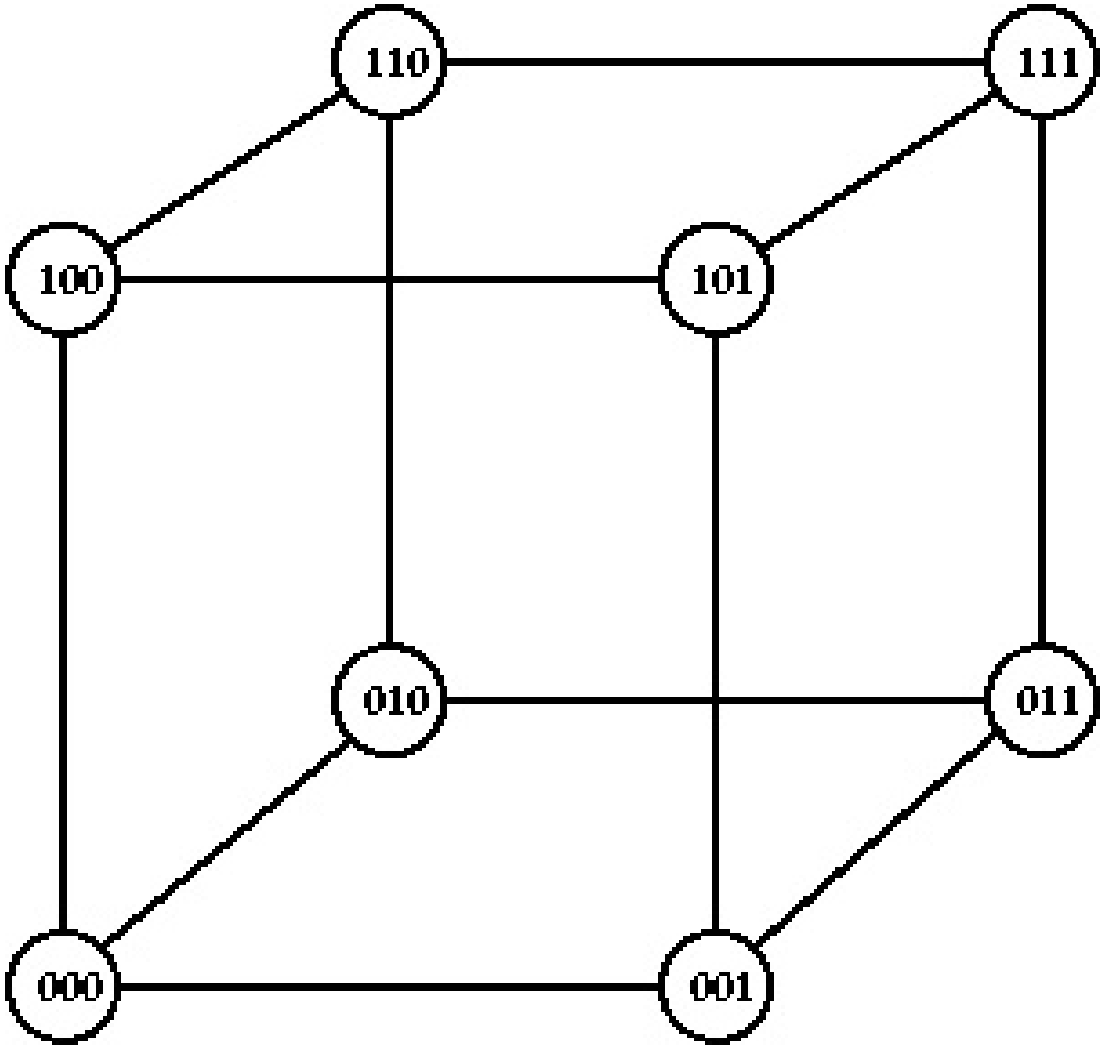
The definition of scheme

Let us suppose that for an optimization problem we represented the possible solutions with a fix length 0-1 sequences. The set of all possible solutions (that is, all 0-1 sequences) is called the search space or solution space. A scheme is a hyper-plane in the solution space, which can be represented using a third character (#, meaning free bit that could be either 0 or 1).

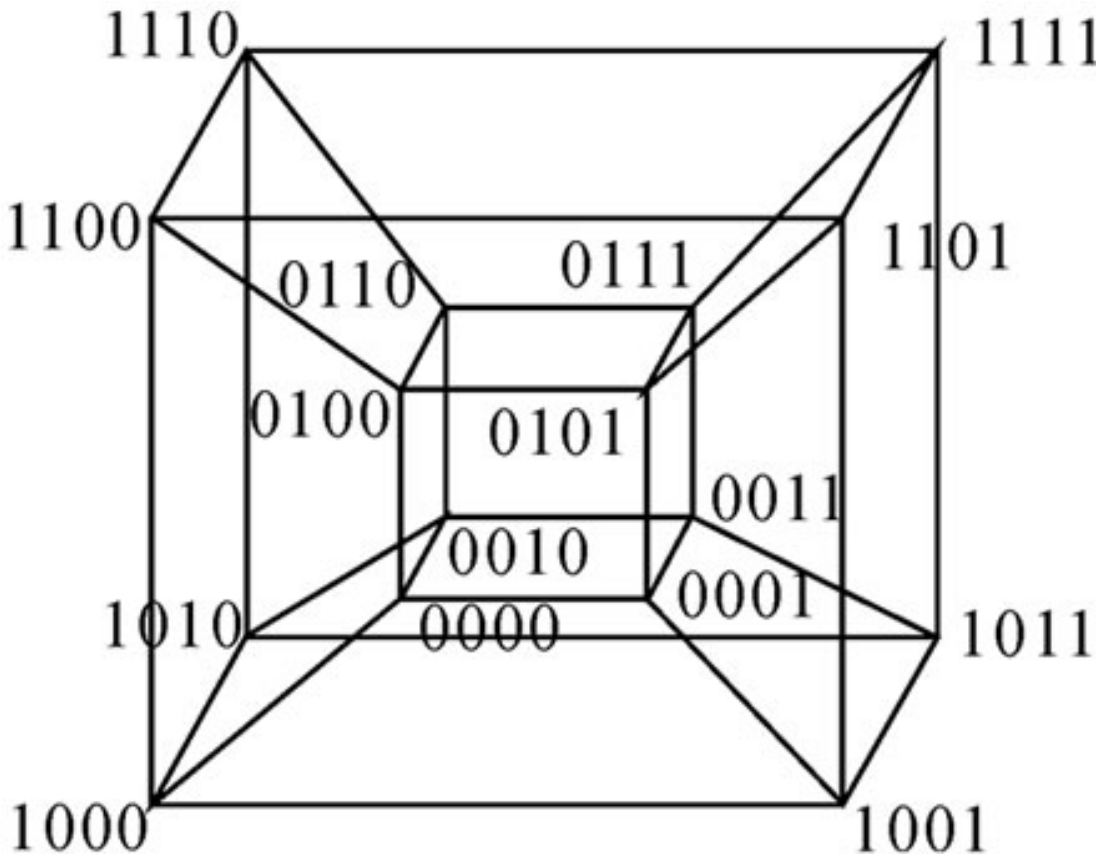
For example, in a search space consisting of 5 length sequences the scheme $H = \#1\#\#0$ fits 8 individuals.

The schemes can be visualized as the vertices/edges/planes/hyper-planes of a hypercube.

3 dimensional cube



4 dimensional cube



Inner parallelism

If the length of the sequence is n, then there are 3^n schemes, and 2^n schemes fit any one individual.

If the population contains μ individuals, then at most $\mu 2^n$ schemes can fit at least one individual in the population. Although typically there aren't that many schemes present at once, Holland proved in 1975 that a genetic algorithm evaluates $\mathbf{O}(\mu^3)$ schemes in one generation.

This property is called inner parallelism, which is one key component of the success of the genetic algorithms.

The properties of schemes

- The fitness of a scheme is the mean fitness of every fitting individual, denoted by $< H >$
- The order of a scheme is the number of the fixed genes, denoted by o(H)
- The defining length of the scheme is the distance of the first and last fixed bits denoted by d(H)

For example if $H = \#1\#\#0$, then its order is 2, its defining length is 3.

Sketch of the proof of the Scheme theorem

Let us suppose, that our algorithm uses roulette-wheel selection, one point crossover and bit-wise mutation. Let us denote the length of the genome (that is the 0-1 sequence) by n.

The probability that scheme H will be destroyed by a one-point crossover is:

$$P_d(H, 1X) = \frac{d(H)}{n-1}$$

The probability that scheme H will be destroyed by the bit-wise mutation is:

$$P_d(H, MUT) = 1 - (1 - P_m)^{o(H)} \sim o(H) \cdot P_m,$$

where P_m denote the probability of the mutation (for each gene).

Sketch of the proof of the Scheme theorem continued

The probability of choosing a scheme as parent depends on the fitness of the scheme. Let us denote the mean of the fitness of individuals in the population by <f>. If in the present population there are $\mu(H,t)$ individuals that scheme H fits, and the population size is ν, then the probability that scheme H will be selected as parent at least once is:

$$P(H,t) = \frac{\mu(H,t) < H >}{\nu < f >}$$

Scheme theorem

Theorem

In the next generation the proportion of scheme H is at least

$$\frac{\mu(H, t+1)}{\nu} \geq \frac{\mu(H, t)}{\nu} \frac{< H >}{< f >} \left(1 - p_c \frac{d(H)}{n-1}\right) (1 - p_m o(H))$$

This shows that the more fit a scheme is, the more it will dominate the population. The inequality also gives an explanation for the slowing convergence rate at the end.

Deceptive problems

We call a problem deceptive, if low order schemes fitting the global optimum have low fitness, while other low order schemes with high fitness don't fit the global optimum at all.

For example let us have 10 items for the backpacking problem, whose weights are [1,2,. . . ,9, 10], and the values are [1, 2 , . . . , 9, 11]. If the capacity is 10, then the optimal packing contains only the last item. However, if we already packed some items, and we are still below the capacity, then we can increase the fitness by packing one more light item (if the sum of the weights is still under the capacity).

In this example we could increase the fitness of packing by going further from the global optimum.

Applying penalties in the fitness

A deceptive problem may be the result of the poor choice of the fitness function.

In this case we might apply some penalty for a packing that surpasses the capacity, instead of giving them 0 fitness. This penalty might even vary in each generation (time dependent penalty) or might depend on the current state of the population (adaptive penalty).

Gray coding

If the solutions are represented as 0-1 sequences meaning a binary integer, then when a new position notation is needed, the structure of the potential solution changes dramatically. So another possible reason for a problem being deceptive is this property of the binary number system.
For example $31 = 011111_2$, but $32 = 100000_2$. If the global optimum is near a power of 2, low order schemes containing a lot of 1-s won't fit it.

If our potential solution is an integer, then it's recommended using Gray coding instead of simply having the binary number, because if two numbers are 'near', then their Gray codes are also 'near'.

Gray coding

The Gray code of a 0-1 sequence is an other 0-1 sequence with the same length.

Let us suppose $n \in \mathbb{N}$ and its binary form is: $b_1 b_2 b_3 \ldots b_k$ (which is a 0-1 sequence) The first bit of the Gray code is b_1, the i. bit is b_{i-1} XOR b_i, where XOR is the exclusive or.

For example:
$n = 11$, then in binary number system $11 = 8 + 2 + 1$ hence $11 = 1011_2$ and then the Gray code is 1110.
If $n = 12$ then, since $12 = 8 + 4$, in the binary number system $12 = 1100_2$ and the Gray code is 1010.

Gray coding II

Definition

Let $b = b_1 b_2 b_3 \dots b_k$ and $c = c_1 c_2 c_3 \dots c_k$ two same length 0-1 sequences. The Hamming-distance of b and c is the number of different bits in b and c.

Theorem

If $n \in \mathbb{N}$ then the Hamming distance of n's and $n+1$'s Gray code is 1.

Proof

If $n = b_1 b_2 \dots b_{k-1} b_k$ in the binary number system, then $n + 1 = b_1 b_2 \dots b_{k-1} b_k + 00 \dots 01_2$.

To calculate the sum of these to base two number we have to start at the end of b. As long as the bits are ones in b the sum will have zeros. When we arrive at the first zero in b (let say in the jth position), then in the sum there will be a 1, after that the bits are identical.

Proof II

When applying the Gray coding to the two numbers, since the beginning of the two numbers are identical, the Gray code will be also identical until the jth position. In the jth position, their Gray code differs, since if we change one bit in a XOR operator, the result will also change. After the jth position, in both numbers there are $10\ldots0$ because after a 10 resp. 01 part in the jth and $j+1$th position all the bits are equal.

We have to finish the proof by observing that if j is the first or last position, then before or after the jth position part is missing, but it doesn't change the fact that the two Gray codes differs in exactly one bit.

Gray code is unique

If two numbers are different, then their Gray codes are also different, because in the position their binary number system form differs, their Gray code also differs.

On the other hand, the Gray coding is invertible: the inverse image of the Gray code $g_1, g_2, g_3 \ldots, g_k$ is $b_1, \ldots b_k$, where $b_1 := g_1$ and $v := b_1$ going onwards:

for i=2:k
if$g_i == 1$ then $v :=\sim v$
$b_i := v$.

The hypothesis of the building blocks

It follows from the scheme theorem, that if two schemes have the same fitness, then the one with the shorter defining length or lower order is more likely to survive the genetic operators.

This observation implies the hypothesis of the building blocks stating that the genetic algorithms working by competing shorter schemes, and using these smaller schemes as building blocks it constructs lengthier schemes fitting the optimum.

Counterexample

We can give a counterexample for the hypothesis of building blocks.

Let us suppose we have the following optimization problem: we have 3-length 0-1 sequences and the fitness function f is the following: f(000)=5, f(111)=4, for the individuals with exactly one 1 in their genome the fitness is 1, and for individuals having two 1's is 3.

Then in any scheme containing at least one 0, changing every 0 to 1 the fitness of the scheme increases, although the global optimum is 000.

Counterexample