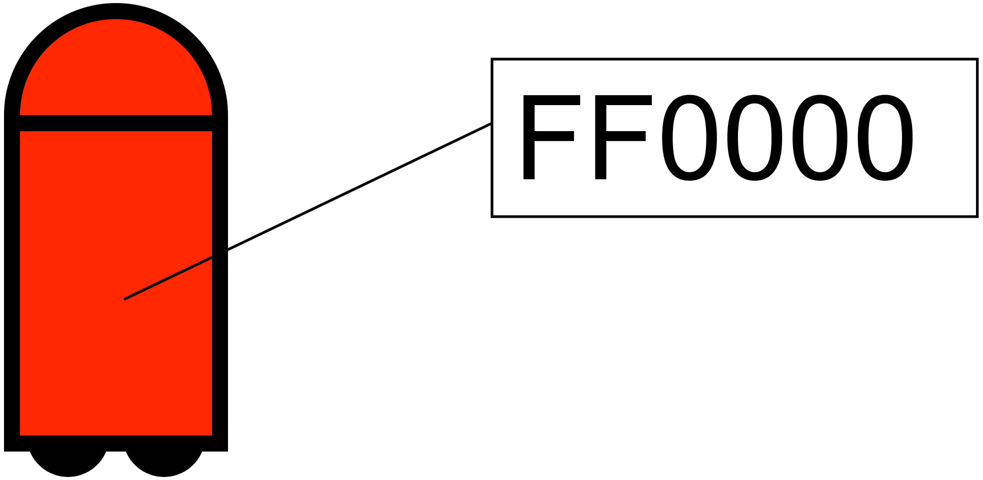
We need something new

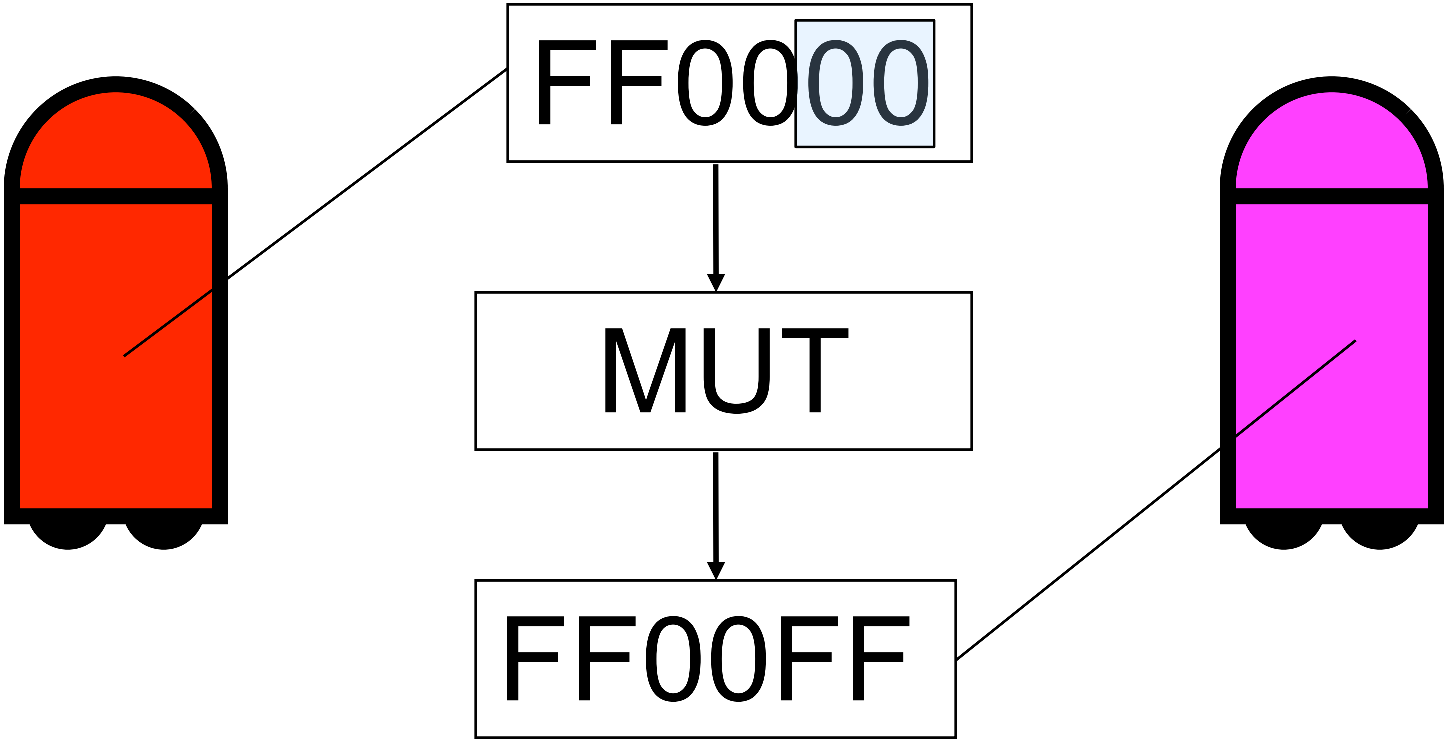
Genetic Algorithms

- GAs revolve around the biological idea of evolution. Individuals taking tiny steps in a random direction, and having to survive in a harsh world.
- Over time a population will improve to become more ‘fit’ according some measure of fitness.

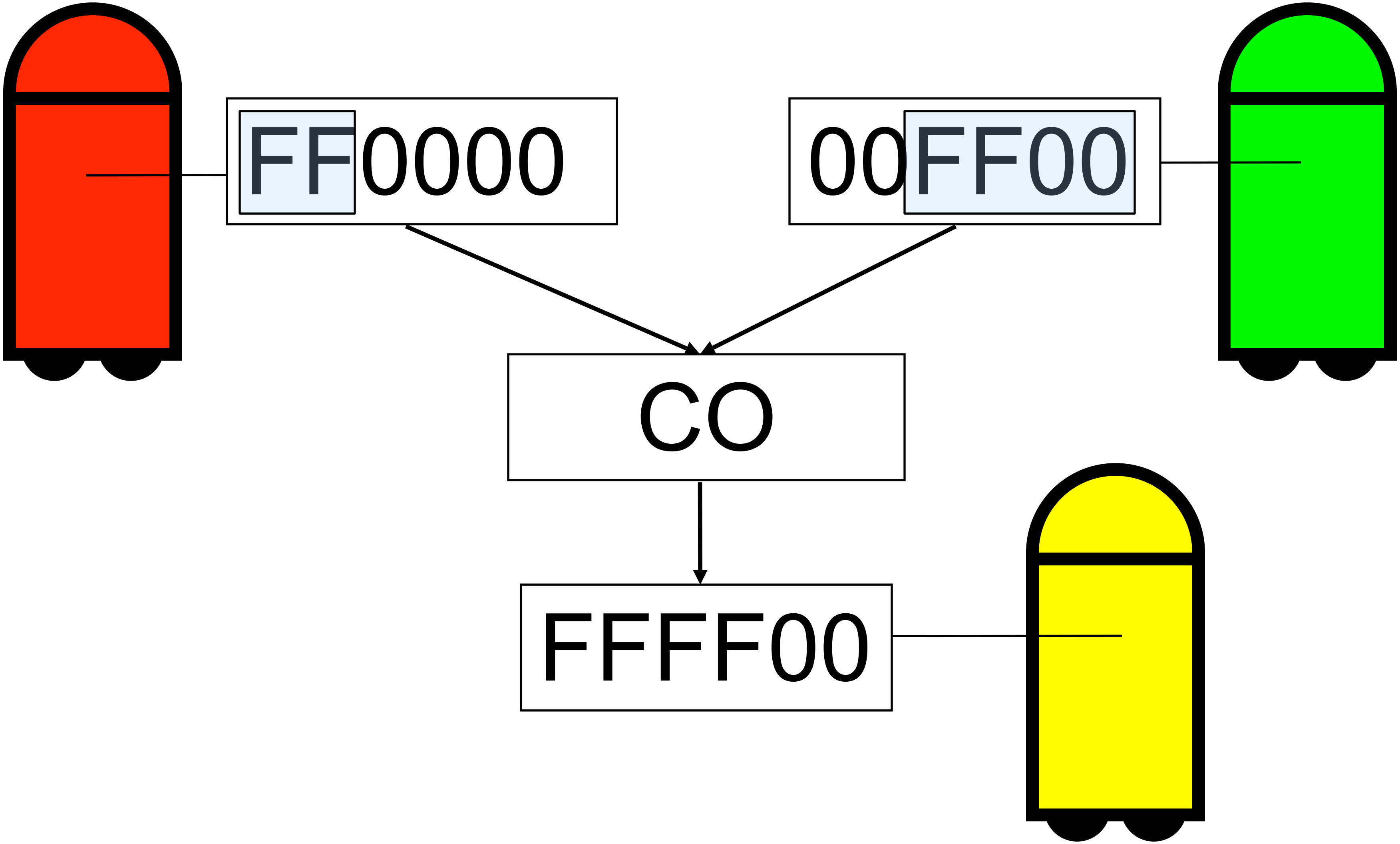
An ‘individual’



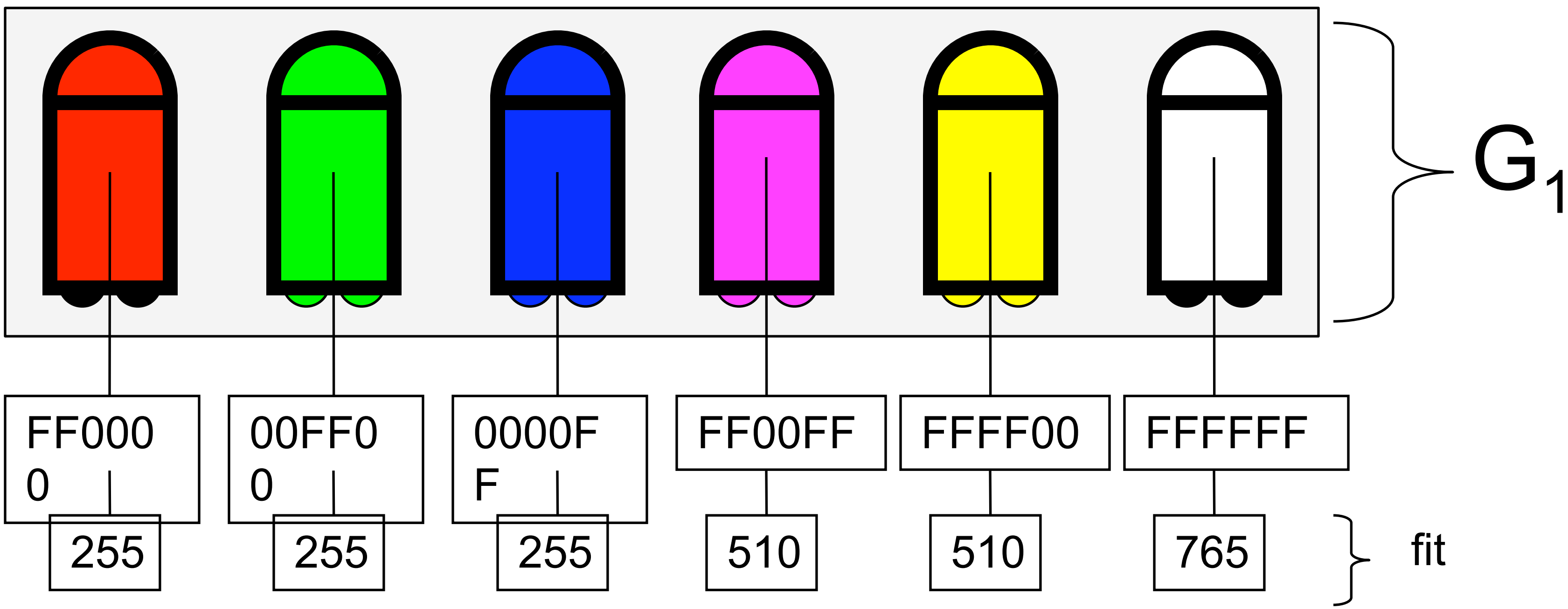
A ‘Mutation’



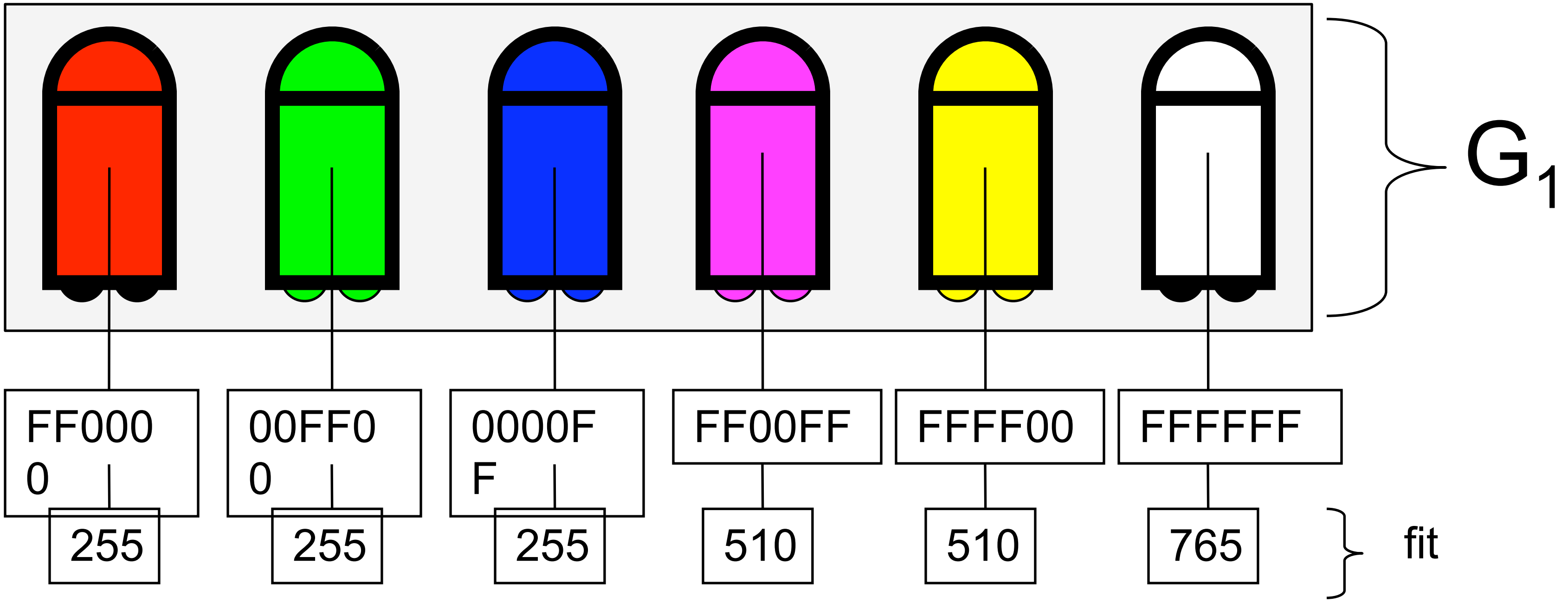
A ‘Crossover’



Fitness

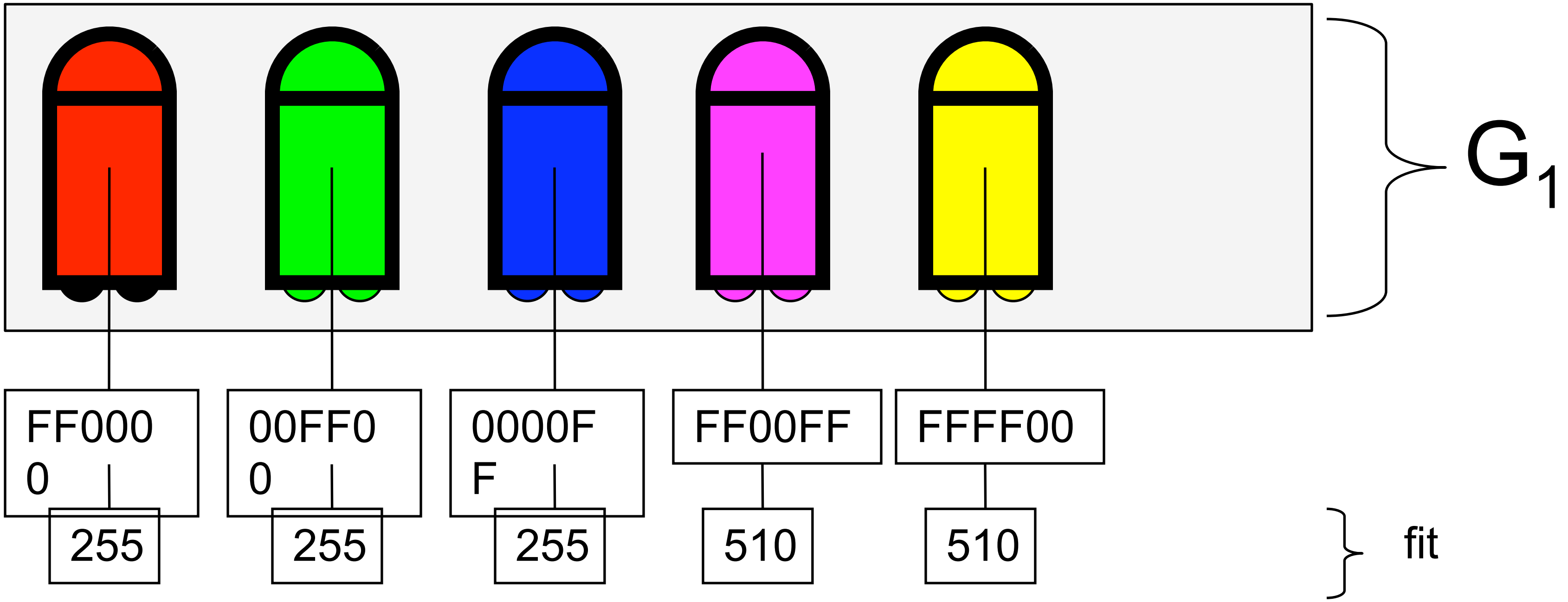

G1
FF000
0
00FF0
0
0000F
F
FF00FF
FFFF00
FFFFFF
255
255
255
510
510
765
fit

Fitness



Suppose we have a constraint that nothing can be 'grey' (ie FFFFFF, 999999, 666666, 333333 are not allowed)

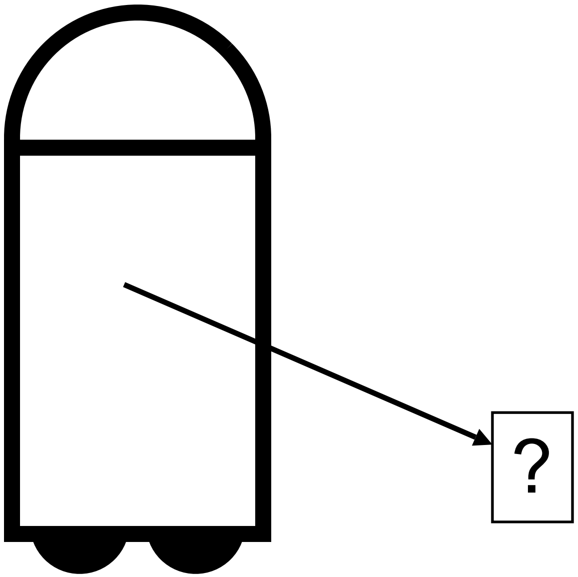
Fitness



Suppose we have a constraint that nothing can be 'grey' (ie FFFFFF, 999999, 666666, 333333 are not allowed)

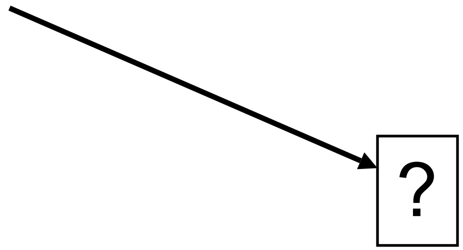
Is this the End?

- Given that (FFFFFF) is not allowed we can simply get rid of it or keep it and fix it

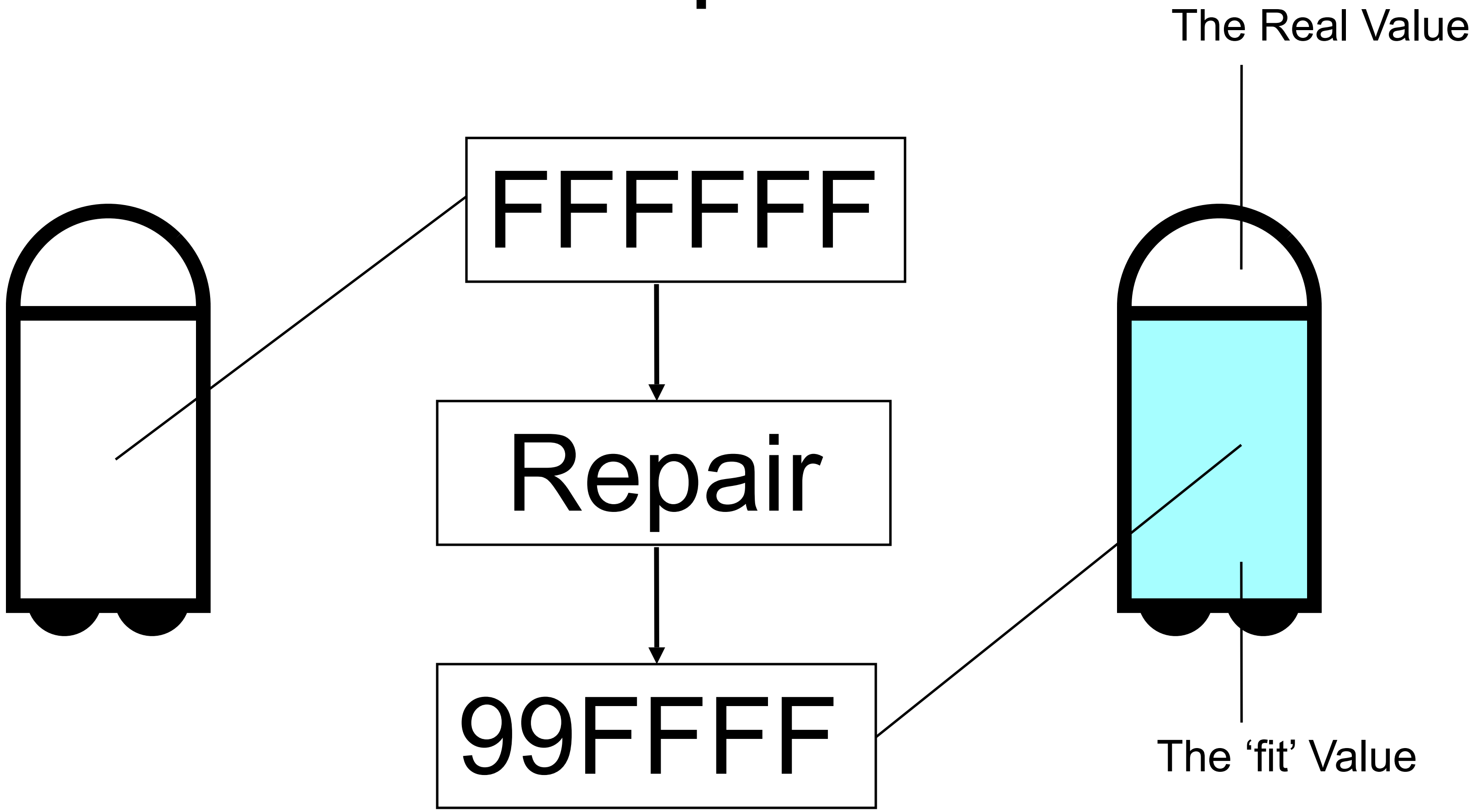


Is this the End?

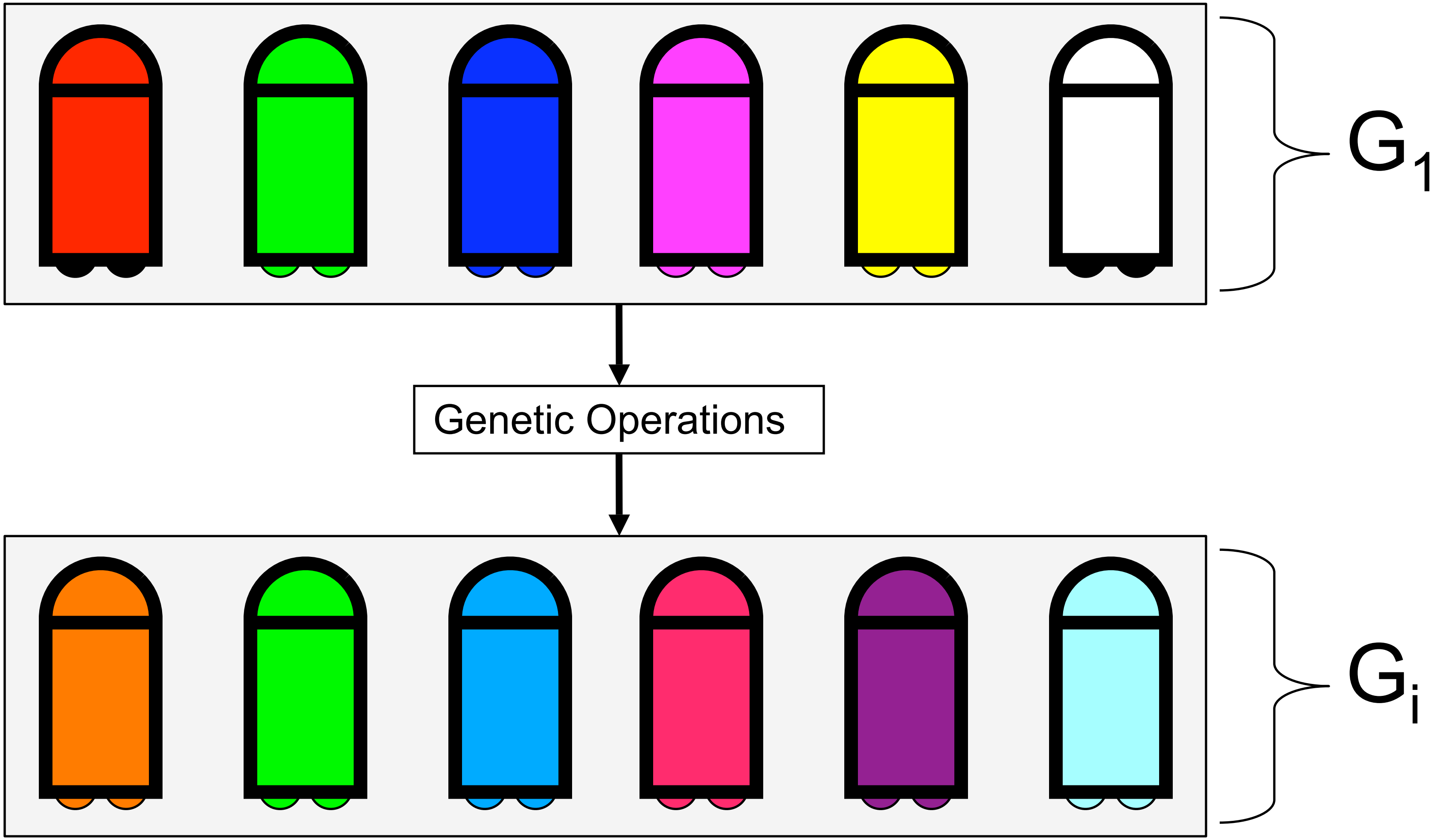
- Given that (FFFFFF) is not allowed we can simply get rid of it or keep it and fix it



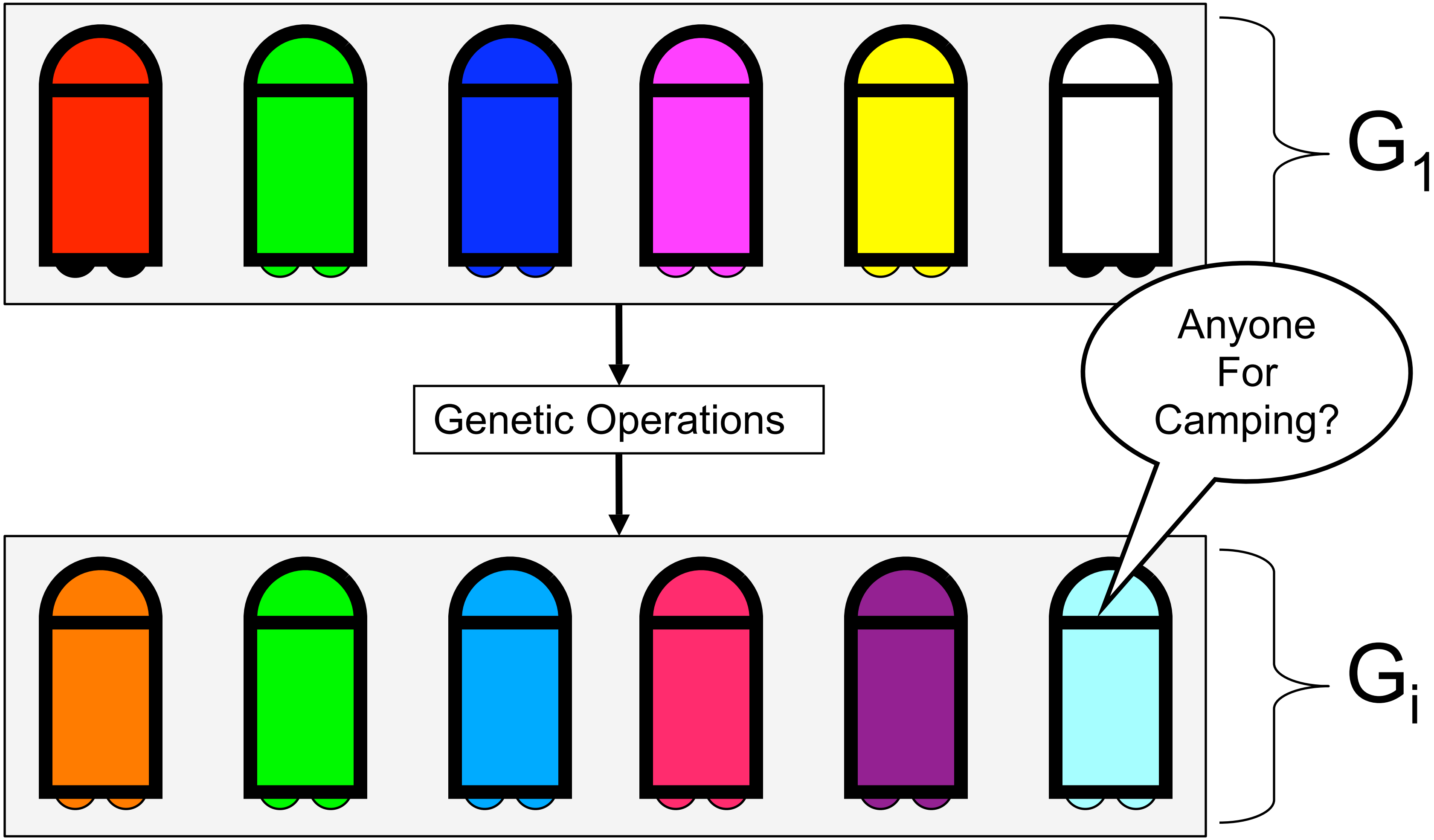
A ‘repair’



Over time we get change



Over time we get change



0-1 Knapsack Problem (as GA)

$c = (2,4,1,1)$

$p = (10,12,2,1)$

$K = 4$

$p_i/c_i = (5,3,2,1)$

$l=5,\ k=3,\ m=6$

0-1 Knapsack Problem

s_0= {(0101) (12), (1100) (10), (0011) (3)}

e_0: *Crossover-to-do* = 0

Repair : (0101) (13) → (0100) (12),

(1100) (22) → (1000) (10)

Since there are no crossovers to do we'll mutate a random individual (here we'll pick (1100) and it change it to (1000) (10)

0-1 Knapsack Problem

s_1= {(0101) (12), (1100) (10), (1000) (10), (0011) (3)}

e_1: *Crossover-to-do* = 2

Now we must do a crossover. Picking (0101) and (0011) and crossing over from position 1 we get (0101)

But wait! (0101) is all ready in our set, so nothing has changed. So in this case s_2=s_1

0-1 Knapsack Problem

s_2= {(0101) (12), (1100) (10), (1000) (10), (0011) (3)}

e_2: *Crossover-to-do* = 1

Now we must do another crossover. Picking (1100) and (0011) and crossing over from position 1 again we get (1011) (13)

Neat! (1011) is the best solution to the problem. There's no way for the search control to know this though so we continue

0-1 Knapsack Problem

s_3= {(1011) (13), (0101) (12), (1100) (10), (1000) (10), (0011) (3)}

e_3: *Crossover-to-do* = 0

A mutation is next, but $|s_3|$ = l now so we have to shrink the population size. The two least fit solutions are (1000) and (0011) so they have to go.

By the rules laid out in the control this doesn't change *Crossover-to-do*

0-1 Knapsack Problem

s_4= {(1011) (13), (0101) (12), (1100) (10)}

e_4: *Crossover-to-do* = 0

Now we can mutate. Picking (1011) and mutating the 3rd bit we get (1001) (11)

This is worse, but that's not a problem.

0-1 Knapsack Problem

s_5= {(1011) (13), (0101) (12), (1001) (11), (1100) (10)}

e_5: *Crossover-to-do* = 2

Next we have another crossover. Picking (0101) and (1100) and crossing-over at 1 we get (0100) (12)

0-1 Knapsack Problem

s_6= {(1011) (13), (0101) (12), (0100) (12), (1001) (11), (1100) (10)}

e_6: *Crossover-to-do* = 1

Since m=6 this is the last state of the search. Excluding the invalid solutions we've found {(1011), (0100), (1001)} and the fittest of these is (1011), which is the result of the search.