N-Boyos Problem!

A genetic issue…

Ignacio Marroquin
Jimmy Johnson

N-Boyos

- Constraints
- nxn squares for n-boys
- http://oeis.org/A000170
    - Sequence of solutions

The Algorithm

- Fitness Function
- Reproduce (Crossover) Operator
- Selection Operator
- Mutation Operator

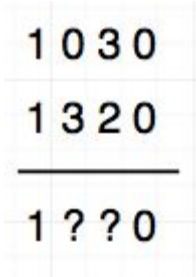
```
def check_clashes(space):
    clashes = 0;
    # calculate row and column clashes
    # just subtract the unique length of array from total lengt
    # [1,1,1,2,2,2] - [1,2] => 4 clashes
    row_col_clashes = abs(len(space) - len(np.unique(space)))
    clashes += row_col_clashes

    # calculate diagonal clashes
    for i in range(len(space)):
        for j in range(len(space)):
            if ( i != j):
                dx = abs(i-j)
                dy = abs(space[i] - space[j])
                if(dx == dy):
                    clashes += 1


    return clashes
```

The Fitness Function

- Fitness = Max_val - clashes

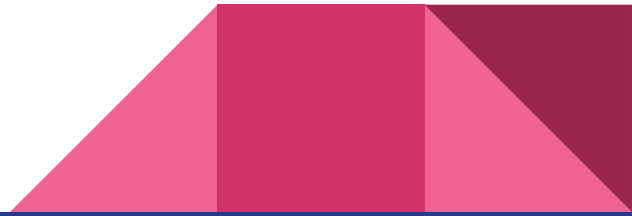
Crossover Operator

- Three way tournament
  - Compare where they are the same
  - If they numbers do not line up add them to the domain.
  - Randomly select a number from domain
    - Remove the number selected to protect uniqueness



Selection Operator

- Delete unfit states (Experiment)
- Using a fitness threshold
- Tournament select
  - Select the fittest out of two.

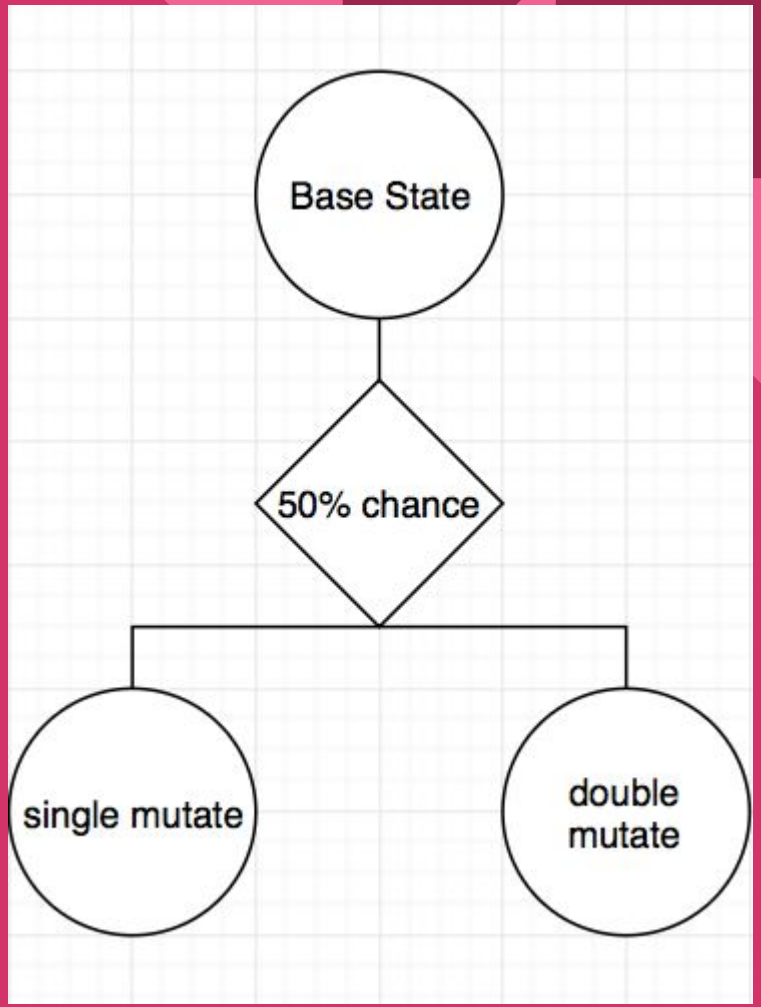


Mutation

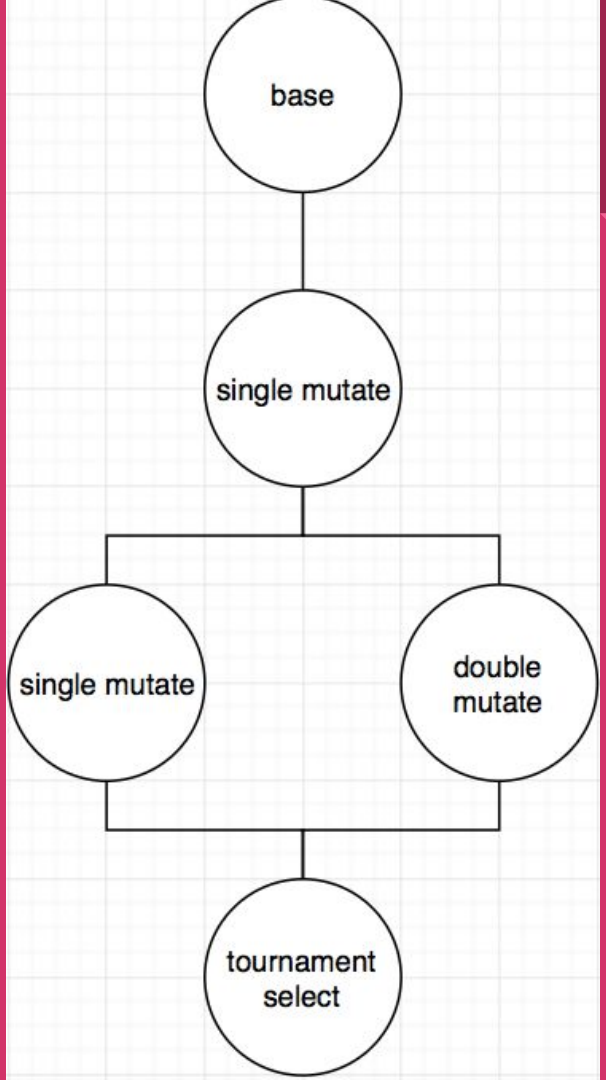
- Slight
- Medium
- Heavy

```
# This will be a bridge for testing out different ways
def eval_mutate(x,y,child,chance=95):
    fit   = child.fit
    fit_a = x.fit
    fit_b = y.fit

    if(r.randint(0,100) > chance):
        if(r.randint(0,100) > 50):
            child.state = slight_mutate(child.state)
            child.fit = cal_fitness(child.state)
            child.mutation = 'Slight'
        elif(r.randint(0,100) > 45):
            child.state = med_mutate(child.state)
            child.fit = cal_fitness(child.state)
            child.mutation = 'Medium'
        else:
            child.state = heavy_mutate(child.state)
            child.fit = cal_fitness(child.state)
            child.mutation = 'Heavy'

    return child
```

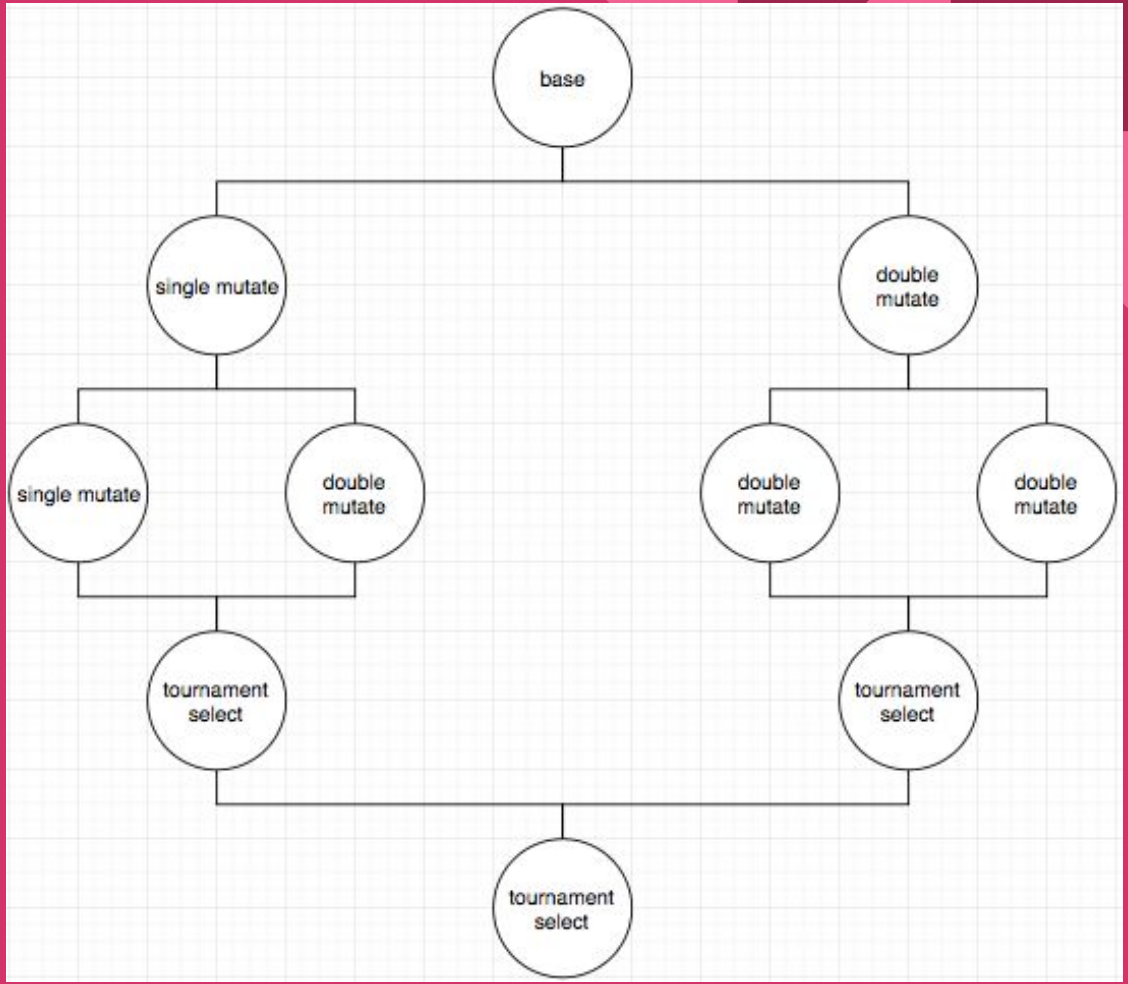
Slight



Medium

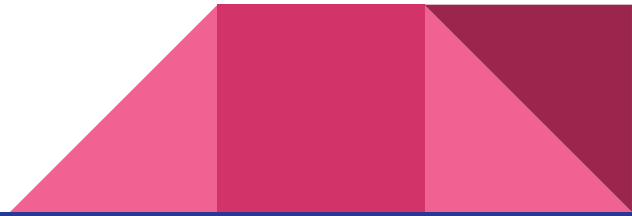

base
single mutate
single mutate
double mutate
tournament select

Heavy


base
single mutate
double mutate
single mutate
double mutate
double mutate
double mutate
tournament select
tournament select
tournament select

Experiment

- Compare time and iteration between different N’s
- Try different Mutation operations
- We noticed that the program starts to take longer after N >= 7



Demo