

Simulation And Optimization Of Traffic Light

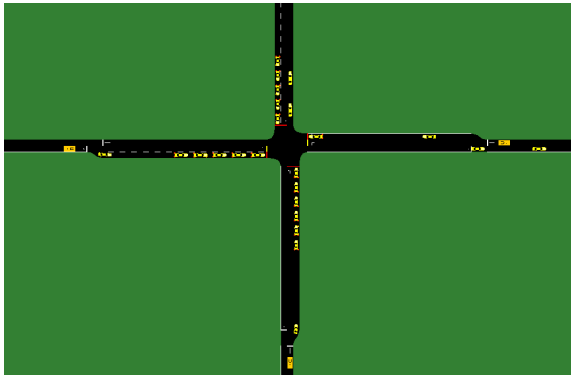
Öztürk Emrah, Nicolai Schwartz, Jan Wenger

January 28, 2020

- 1 Introduction
- 2 Scenario/Program
- 3 Optimization Algorithm
- 4 Experiments/Results
- 5 Summary

optimization of fairness, stop time and number of stops

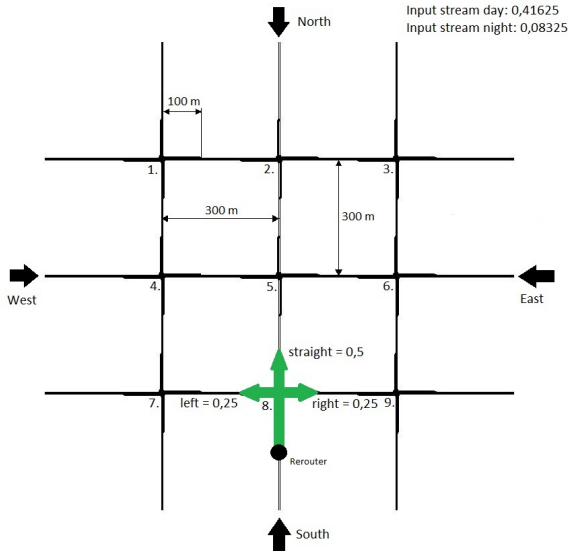
$$func = \frac{g_1 fairness + g_2 stop_time + g_3 number_stops}{3} \rightarrow \min$$

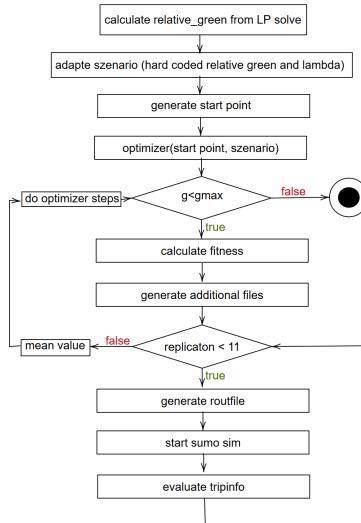


Simplifications

- simplifications of software SUMO
- velocity, gap between cars and acceleration same
- one constant input stream from every geographic direction (N,S,W,E)
- perfection = 1, same turn probabilities (0.5 straight ahead, 0.25 left and right)
- one type of car, non intelligent, dynamic rerouteing
- exponential distribution for input streams
- stop simulation after 3600 simulation steps

- 1 Introduction
- 2 Scenario/Program**
- 3 Optimization Algorithm
- 4 Experiments/Results
- 5 Summary





- 1 Introduction
- 2 Scenario/Program
- 3 Optimization Algorithm**
- 4 Experiments/Results
- 5 Summary

Differential Evolution

Differential evolution

```
1: population  $\leftarrow$  initialization
2: while  $g < G_{max}$  do
3:   for individual  $x_i$  in population do
4:      $v_i = \text{mutation}(x_i, \text{population}, F)$ 
5:      $u_i = \text{crossover}(x_i, v_i, CR)$ 
6:     if  $\text{function}(u_i) < \text{function}(x_i)$  then
7:        $x_i = u_i$ 
8:     end
9:      $g = g + 1$ 
10:  end
11: end
```

NSGA-II

```

1: population  $\leftarrow$  initialization
2: while  $g < G_{max}$  do
3:   for  $i$  in  $\frac{population}{2}$  do
4:      $p_1, p_2 = \text{tournament\_selection}(P_t)$ 
5:      $q_1, q_2 = \text{crossover}(p_1, p_2)$ 
6:      $q_1 = \text{mutations}(q_1)$ 
7:      $q_2 = \text{mutations}(q_2)$ 
8:      $Q_t = Q_t \cup (q_1, q_2)$ 
9:   end
10:   $R_t = R_t \cup (Q_t)$ 
11:   $R_t = \text{fast\_non\_dominated\_sort}(R_t)$ 
12:   $P_t = \text{crowding\_distance\_sorting}(R_t, P_t.size)$ 
13:   $g = g + 1$ 
14: end
    
```

Conjugate Gradient Descent

```

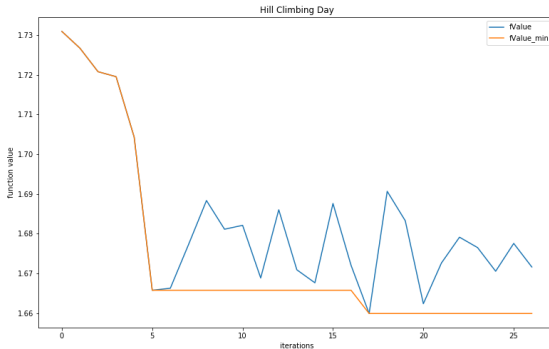
1:  $x, d \leftarrow \text{initialization}$ 
2: while  $\|\nabla f(x)\| > \epsilon$  or  $\|\hat{\eta}d\| < \epsilon$  do
3:    $\text{grad} = \text{numGrad}(x)$ 
4:    $d = -\text{grad} + \frac{\|\text{grad}\|^2}{\|\text{grad}_{old}\|^2} d$ 
5:   if  $\frac{\text{grad}d}{\|\text{grad}_{old}\| \|d\|} > -\alpha$  then
6:      $d = -\text{grad}$ 
7:   end
8:    $\hat{\eta} = \text{linsearch}(f(x, d))$ 
9:    $x_{old} = x$ 
10:   $\text{grad}_{old} = \text{grad}$ 
11:   $x = x + \hat{\eta}d$ 
12: end
    
```

Hill Climbing

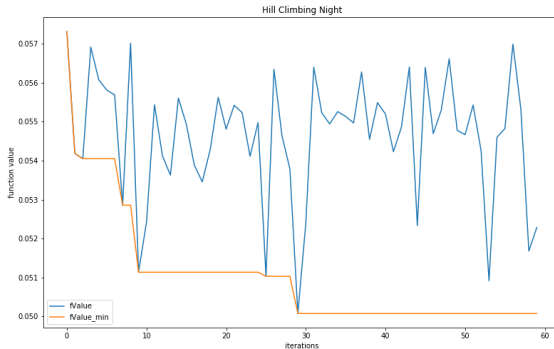
```
1:  $x_{start} \leftarrow$  initialization
2: while  $fe < \#FE_{max}$  do
3:   for  $d$  in Dim do
4:      $z[d] = step$ 
5:      $y_{1,2} = x \pm step$ 
6:      $f = function(y)$ 
7:      $fe = fe + 1$ 
8:   end
9:   if  $function(y_d) < function(x)$  then
10:     $x = y_d$ 
11:   end
12:   else
13:     $step = \frac{step}{2}$ 
14:   end
15: end
```

- 1 Introduction
- 2 Scenario/Program
- 3 Optimization Algorithm
- 4 Experiments/Results**
- 5 Summary

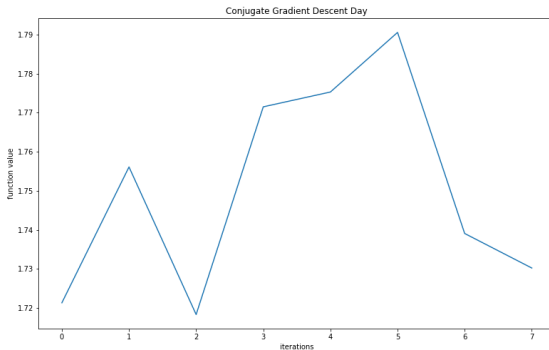
NSGA-II
parameter
epsilon = 0.5
*popsiz*e = 30
gen = 20



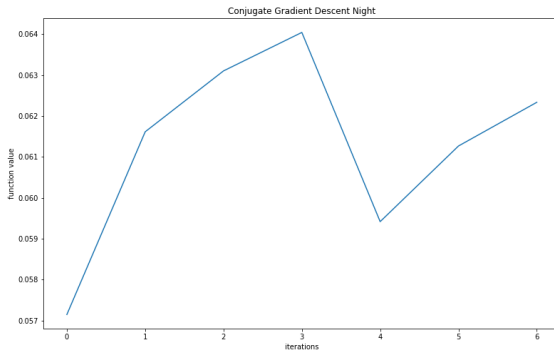
Hill-Climbing
parameter
 $\epsilon = 0.5$



Hill-Climbing
parameter
 $\epsilon = 0.1$



Cgd
parameter
 $\epsilon = 0.5$
 $\alpha = 0.1$
 $\epsilon = 0.1$
 $\eta = 10$
 $h = \text{eps}$



Cgd
parameter
 $\epsilon = 0.1$
 $\alpha = 0.1$
 $\eta = 10$
 $h = \epsilon$

- 1 Introduction
- 2 Scenario/Program
- 3 Optimization Algorithm
- 4 Experiments/Results
- 5 Summary

Summary

- Problems:
 - time consuming (calculation time)
 - program crash (sumo)
 - teleportation of cars
- Reduction of the cost function is observable
- No possibility to check for correctness
- Less function evaluations → less calculation time needed
- Conjugate gradient descent problem with line-search
(probability based rout file generation)