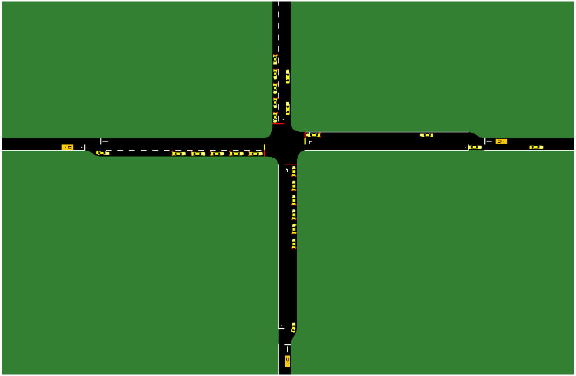
Simulation And Optimization Of Traffic Light

Öztürk Emrah, Nicolai Schwartze, Jan Wenger

January 28, 2020

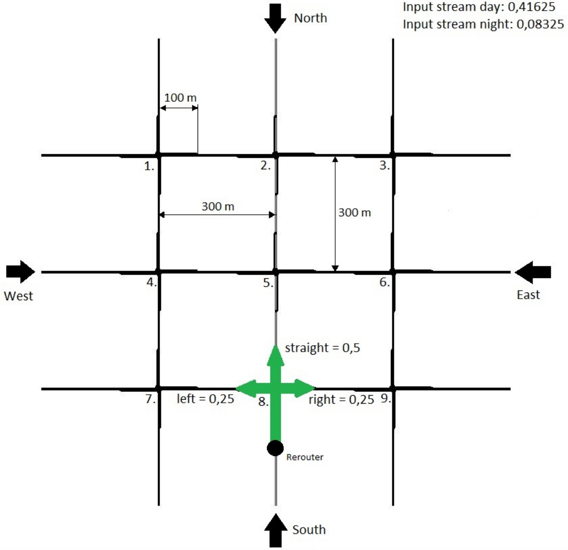
optimization of fairness, stop time and number of stops

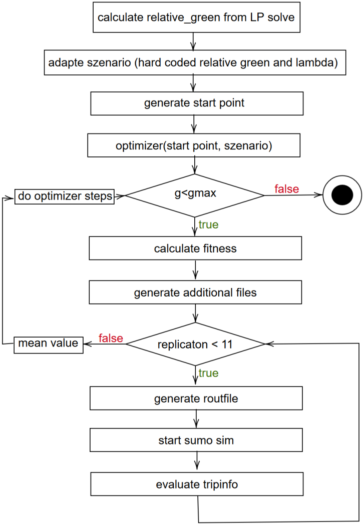
$$func = \frac{g_1 fairness + g_2 stop\_time + g_3 number\_stops}{3} \rightarrow \min$$



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


North
Input stream day: 0,41625
Input stream night: 0,08325
100 m
1.
2.
3.
300 m
300 m
West
4.
5.
6.
East
straight = 0,5
7.
left = 0,25
8.
right = 0,25
9.
Rerouter
South


calculate relative_green from LP solve
adapte szenario (hard coded relative green and lambda)
generate start point
optimizer(start point, szenario)
do optimizer steps
g<gmax
false
true
calculate fitness
generate additional files
mean value
false
replicaton < 11
true
generate routfile
start sumo sim
evaluate tripinfo

Differential Evolution

Differential evolution

population $\leftarrow$ *initialization*
while $g < G_{max}$ **do**
  for individual x_i **in** population **do**
    v_i = mutation(x_i, population,F)
    u_i = crossover(x_i, v_i,CR)
    if function(u_i) < function(x_i) **then**
      $x_i = u_i$
    end
  end
  $g = g + 1$
end

NSGA-II

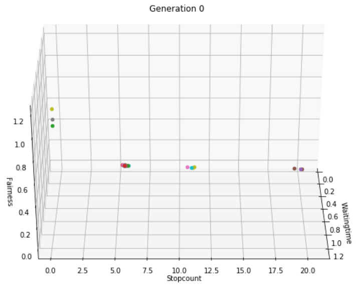
population $\leftarrow$ *initialization*
while $g < G_{max}$ **do**
  for i **in** $\frac{population}{2}$ **do**
    $p_1, p_2 =$ tournament_selection(P_t)
    $q_1, q_2 =$ crossover(p_1, p_2)
    $q_1 =$ mutations(q_1)
    $q_2 =$ mutations(q_2)
    $Q_t = Q_t \cap (q_1, q_2)$
  end
  $R_t = R_t \cap (Q_t)$
  $R_t =$ fast_non _dominated _sort(R_t)
  $P_t =$ crowding_distance _sorting$(R_t, P_t.size)$
  g = g + 1
end

Conjugate Gradient Descent

x,d $\leftarrow$ initialization
while $||\nabla f(x)|| > \epsilon$ **or** $||\hat{\eta}\mathrm{d}|| < \epsilon$ **do**
    grad = numGrad(x)
    d = -grad + $\frac{||grad||^2}{||grad_{old}||^2 d}$
    if $\frac{gradd}{||grad_{old}||||d||} > -\alpha$ **then**
        d = -grad
    end
    $\hat{\eta} = linsearch(f(x, d))$
    $x_{old} = x$
    $grad_{old} = grad$
    $x = x + \hat{\eta} d$
end

Hill Climbing

$x_{start} \leftarrow$ initialization
while $fe < \#FE_{max}$ **do**
  for d **in** Dim **do**
    $z[d] = step$
    $y_{1,2} = x \pm step$
    $f = function(y)$
    $fe = fe + 1$
  end
  if function(y_d)<function(x) **then**
    $x = y_d$
  end
  else
    step $= \frac{step}{2}$
  end
end

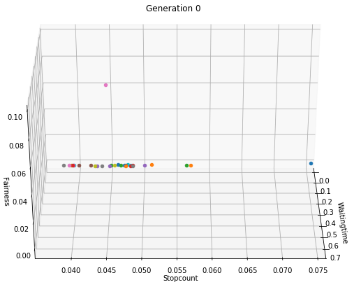


NSGA-II parameter

$epsilon = 0.5$

$popsize = 30$

$gen = 20$



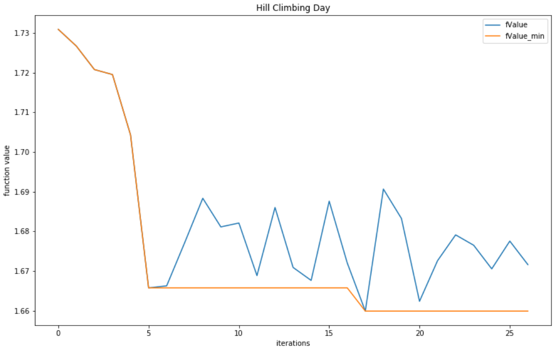
NSGA-II

parameter

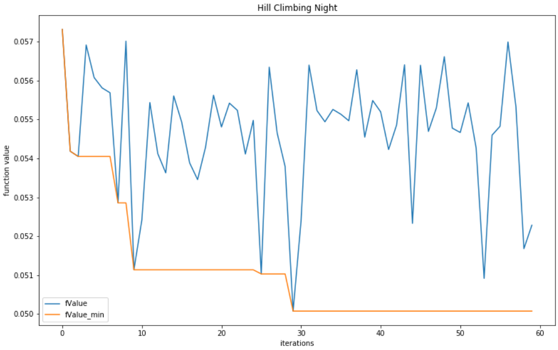
$epsilon = 0.1$

$popsize = 30$

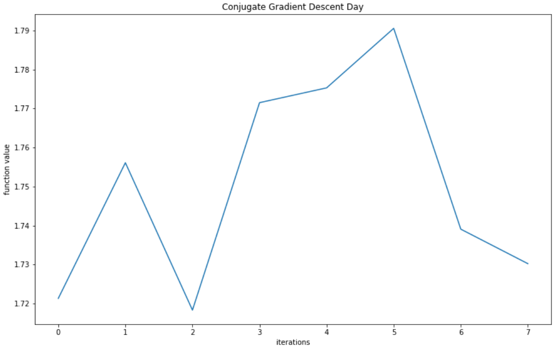
$gen = 15$

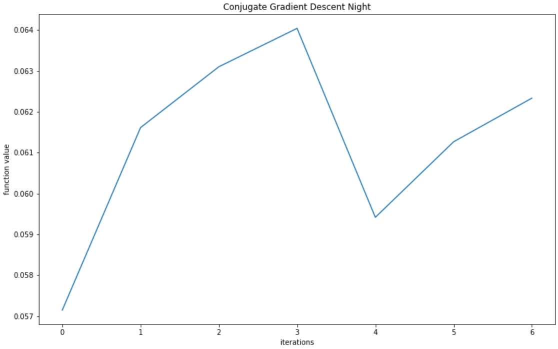


Hill-Climbing parameter $epsilon = 0.5$



Hill-Climbing parameter $epsilon = 0.1$

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



Cgd parameter

$epsilon = 0.1$

$\alpha = 0.1$

$\epsilon = 0.1$

$\eta = 10$

$h = eps$

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 $\rightarrow$ less calculation time needed
- Conjugate gradient descent problem with line-search (probability based rout file generation)