Comparison Between DASH and ACTR

See the rest here: https://github.com/DannyWeitekamp/HLAHTOI/tree/master/assignment1 (https://github.com/DannyWeitekamp/HLAHTOI/tree/master/assignment1)

In [7]:
```
import numpy as np
import pandas as pd
```

In [8]:
```
def gen_learner_history(now,times,correct,alpha=0.0,delta=0.0,windows=[1/24,1,7,3
0]):
    times = np.array(times)
    correct = np.array(correct)
    elapse = now-times
    windows = [0.0] + [86400.0*x for x in windows] + [float('inf')]
    lh = {}
    for i in range(len(windows)-1):
        upper = windows[i+1]
        lower = windows[i]
        in_win = np.logical_and(elapse > lower,elapse <= upper)
        c_in_win = in_win * correct

        lh['n_w'+str(i)] = in_win.sum()
        lh['c_w'+str(i)] = c_in_win.sum()
    lh["elapses"] = elapse
    lh["alpha"] = alpha
    lh["delta"] = delta
    return lh

def day(x):
    return x * 86400.0
def hour(x):
    return x * 3600
def mins(x):
    return x * 60
```

Fitting The DASH Model

Because DASH doesn't come with a description of parameter values, I had to fit a model to data. I used some data that Theo used in his study; preprocessed data from the assistments17 dataset. See the other two notebooks in this github for how I processed the data further for this purpose. I tried fitting the data with pyTorch, but it didn't converge, so I tried again by using with the glmer package in R (this one worked better). I made a phi and psi fixed effects and had random effects for the student and kc intercepts (I also did not use a global intercept). I include the fit parameters for both models here.

In [9]:
```
#---Fit with torch---
#phi tensor([-0.1015,  0.1419,  0.1439,  0.1933,  0.0000])
#psi tensor([-0.1941,  0.1973,  0.0245,  0.0347,  0.0000])

#---Fit with glmer (in R)----
#    ltc_0      lop_0      ltc_1      lop_1      ltc_2      lop_2      ltc_3
lop_3      ltc_4  lop_4
# 0.038408  -0.011849   0.048038  -0.032408   0.033345  -0.024711  -0.003275   0.
005272  -0.043829  0.033895


window_profiles = [{
    'scale' : 1/24,
    'phi' : .038408,
    'psi' : -0.011849
},
{
    'scale' : 1,
    'phi' : 0.048038,
    'psi' : -0.032408
},
{
    'scale' : 7,
    'phi' : 0.033345,
    'psi' : -0.024711
},
{
    'scale' : 30,
    'phi' : -0.003275,
    'psi' : 0.005272
},
]

def sigmoid(x):
    return 1/(1 + np.exp(-x))

def DASH_activation(lh,wps):
    s = 0
    for i,wp in enumerate(wps):
        s += wp["phi"]*np.log(1+lh["c_w"+str(i)])-wp["psi"]*np.log(1+lh["n_w"+str
(i)])
    #print(s*10)
    s*=10
    return lh['alpha'] - lh['delta'] + s

def DASH(lh,wps):
    x = DASH_activation(lh,wps)
    return 1.0/(1 + np.exp(-x))

def whitehill_activation(lh):
    B_t = (lh['alpha'] - lh['delta']) #Best Guess
    return np.log(np.sum(np.power(lh['elapses']/3600.0,-.5)))+B_t

def whitehill(lh):
    x = whitehill_activation(lh)
    return 1.0/(1 + np.exp(-x))
```

Comparison Between DASH and Whitehill Implementation of(ACT-R)

I compare predictions of DASH vs Whitehill's ACT-R implementation on three different studying patterns:

1. Crammed: A little bit of study 2 days before, lots of study 30 minutes before the test
2. Spaced: Study sessions are spaced over different days
3. Massed: Lots of Study 2 days before the test

In [10]:
```
cram_h = [
    day(0)+hour(0)+mins(1),
    day(0)+hour(0)+mins(2),
    day(0)+hour(0)+mins(3),
    day(2)+hour(4)+mins(31),
    day(2)+hour(4)+mins(32),
    day(2)+hour(4)+mins(33),
    day(2)+hour(4)+mins(34),
    day(2)+hour(4)+mins(35),
    day(2)+hour(4)+mins(36),
    day(2)+hour(4)+mins(37),
    day(2)+hour(4)+mins(38),
]
cram_c = [0,0,1,0,0,1,1,0,1,1,1]


spaced_h = [
    day(0)+hour(0)+mins(1),
    day(0)+hour(0)+mins(2),
    day(0)+hour(0)+mins(3),
    day(1)+hour(4)+mins(31),
    day(1)+hour(5)+mins(32),
    day(1)+hour(6)+mins(33),
    day(1)+hour(7)+mins(34),
    day(2)+hour(1)+mins(35),
    day(2)+hour(2)+mins(36),
    day(2)+hour(3)+mins(37),
    day(2)+hour(4)+mins(38),
]
spaced_c = [0,0,1,0,0,1,1,0,1,1,1]

massed_h = [
    day(0)+hour(0)+mins(1),
    day(0)+hour(0)+mins(2),
    day(0)+hour(0)+mins(3),
    day(0)+hour(1)+mins(31),
    day(0)+hour(1)+mins(32),
    day(0)+hour(1)+mins(33),
    day(0)+hour(1)+mins(34),
    day(0)+hour(1)+mins(35),
    day(0)+hour(1)+mins(36),
    day(0)+hour(1)+mins(37),
    day(0)+hour(1)+mins(38),
]
massed_c = [0,0,1,0,0,1,1,0,1,1,1]
```

Results

Interestingly ACT-R predicts that cramming is the best, but DASH predicts that spacing is the best. Values are printed out here as probabilities (from 0 to 1).

In [11]:
```
now = day(2)+hour(5)
cram_lh = gen_learner_history(now,cram_h,cram_c,alpha=.05,delta=.1)
spaced_lh = gen_learner_history(now,spaced_h,spaced_c,alpha=.05,delta=.1)
massed_lh = gen_learner_history(now,massed_h,massed_c,alpha=.05,delta=.1)

print("DASH Predictions")
print("Crammed\t:", DASH(cram_lh,window_profiles))
print("Spaced\t:", DASH(spaced_lh,window_profiles))
print("Massed\t:", DASH(massed_lh,window_profiles))
print("---------------------------------")
print("Whitehill Predictions")
print("Crammed\t:", whitehill(cram_lh))
print("Spaced\t:", whitehill(spaced_lh))
print("Massed\t:", whitehill(massed_lh))
```

```
DASH Predictions
Crammed : 0.8133953765694286
Spaced  : 0.911409320752136
Massed  : 0.7708132621017548
---------------------------------
Whitehill Predictions
Crammed : 0.9236686430193123
Spaced  : 0.8244147745741137
Massed  : 0.5923833532536381
```

In [ ]: