This file contains most of my working through ILP design, these are all rough notes so some of it may be difficult to understand. I am always available to help out at ahsansbutt@hotmail.com.

I start off with how basic experiments can be described in ILP and end with the experiments I performed and the times it took for different modifications to ILP-solvers for those experiments as compared to unigen.

Additionally I started to make some changes to implement ILP but it is not complete. My plan was to implement ILP as a separate strategy and create a class function to replicate “build\_backend\_request” -> “build\_ILP\_solver” for each kind of Block subclass. I believe I have implemented the basics in factors and some constraints, but the complex windows, constraints and the sampling strategy itself need to be changed, and ofcourse lots of testing is required.

This file is a part of notes in the repository, I will add other files to the same directory in this branch to allow you to see how I ran my experiments. Most of the test cases will be almost impossible to understand; once again reach out if you need help.

Although I started off testing with gurobi under a student license, gurobi is not a free library, and we cannot add that to sweetpea. So I have switched to Pulp. You can find a basic implementation of Pulp in the notes.

To make changes to the repo you will need access from Prof Matthew Flatt.

To upload a new version of sweetpea to Pip you will need access from Ben Draut.

Some important links;

<https://coin-or.github.io/pulp/>

Goodluck!!

F: A: a0 a1

F: B: b0 b1

T1:

1 2 3 4

[T1, A, a0], [T1, A, a1], [T1, B, b0], [T1, B, b1]

Variables: {[trial, factor\_name, value]}

sum(1, 7) LT 2

sum(7, 13) LT 2

sum(13, 19) LT 2

< 2

(1, 7) LT 2

<3  
(1, 7, 13) LT 2

sum(variables) LT k

At most 2 in a row among {1, 7, 13, 21, 26, 29}

x1 + x7 + x13 < 3

x7 + x13 + x21 < 3

…

At least 2 in a row among {1, 7, 13, 21, 26, 29}

110011

1. sum({1, 7, 13, 21, 26, 29}) >= k
2. x1 and x13 -> x7
3. X7 and x21 -> x13

At least 3 in a row among {1, 7, 13, 21, 26}

1. sum({1, 7, 13, 21, 26}) >=3 -> need 3 ones
2. X1 and x13 -> x7 -> 101xx not allowed
3. X7 and x21 -> x13 -> x101x not allowed
4. X13 and x26 -> x21 -> xx101 not allowed

3

{1, 2, 3, 4, 5, 6, 7, 8, 9,10}

111000111

import gurobipy as gp

from gurobipy import GRB

import numpy as np

import scipy.sparse as sp

try:

# Create a new model

m = gp.Model("mip1")

# Create variables

b = np.array([1]\*24)

block = m.addMVar(shape=24, vtype=GRB.BINARY, name="block")

# b = sp.csr\_matrix((b, 1), shape=(24, 1))

# Set objective

c = np.random.gumbel(0, 1, size=(1,24))

w = np.array([1]\*24)

w[2] = 23

G = c+np.log(w)

# option 1: G = c + w

# option 2: G = c + np.log(w)

# option 3: G = (c + w)

# G = G / sum(G) ?

# option 4: w = w/sum(w)

# G = c + w

m.setObjective(G @ block, GRB.MAXIMIZE)

# Set contraint

m.addConstr(b @ block == 1, name="c")

# Set to find multiple solutions

m.Params.PoolSearchMode=2

m.Params.PoolSolutions=26

# Optimize model

m.optimize()

m.Params.SolutionNumber=0

for v in m.getVars():

print('%s %g' % (v.varName, v.x))

except gp.GurobiError as e:

print('Error code ' + str(e.errno) + ': ' + str(e))

except AttributeError:

print('Encountered an attribute error')

B1,b2,b3,b4,b5,b6,b7,b8

10000000

01000000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Color | Text | Res-trans | Congruency | Response |
| X1, x2, x3, x4 | X5, x6, x7, x8 | X9, x10 | X11, x12 | X13, x14, x15, x16 |
| X17, x18, x19, x20 | X21, x22, x23, x24 | X25, x26 | X27, x28 | X29, x30, x31, x32 |

………………………

………...times 33

B1: x1, x5, x9

Crossing: color text res-trans

|  |  |
| --- | --- |
| X1, x2, x3, x4 | * Consistency: X1+x2+x3+x4 = 1 -- valid for all factors |
| X5, x6, x7, x8 |  |
| X9, x10 |  |
| X11, x12 | * x11(con)/x12(incon) * X11 <-> Some combination of factors color and text (C, T) * (x1^x5)v(x2^x6)v(x3^x7)v(x4^x8) ⇔ x11 * y1=>x11 * x11=>y1 * Quite a few more variables?   2y1-x1-x2 |
| X13, x14, x15, x16 |  |

X1,x2

1,0

Y1,y2

1,1

1,0

Max(x1,x2,x3,x4,x5,x6,x7,x8)

Normal Sweetpea results:

Color: red, blue

Text: red blue

No constraints

24 unique solutions

|  |  |
| --- | --- |
| Requested | Unique solutions |
| 50 | 21 |
| 100 | 23 |
| 150 | 24 |
| 125 | 23 |
| 140 | 24 |
| 135 | 24 |
| 130-131 | 23-24 |

Split results

|  |  |
| --- | --- |
| Requested | Unique solutions |
| 25 25 | 16 |
| 50 50 | 16 |
| 100 100 | 23 |
| 130 130 | 31 |
| 131 131 | 32 |
| 130x100 | 31 |

Empty cnf, only 5 variables declared. 32 unique solutions, tested upto 100 times

|  |  |
| --- | --- |
| Requested | Unique solutions |
| 100 | 26 |
| 150 | 29 |
| 116-117 | 29-30 |
| 120-121 | 30-31 |
| 122-123 | 31-32 |

The value missing does not change. It is one unique solution that is the last to be found.

So 100 calls for 122 samples will still miss a unique sample even when combined.

|  |  |
| --- | --- |
| Requested(empty Cnf) 32 solutions | Unique solutions(mean, max, min, std dev) |
| 25 | 17.83, 22, 14, 1.6556267695347282 |
| 50 | 25.16, 29, 20, 1.932459572668986 |
| 75 | 29.08, 32, 25, 1.5144636014114041 |
| 100 | 30.55, 32, 28, 1.0897247358851683 |
| 125 | 31.43, 32, 28, 0.7649182962905253 |
| 150 | 31.78, 32, 30, 0.4377213725647859 |

|  |  |
| --- | --- |
| Requested(sweetpea setting-no constraints) 24 solutions | Unique solutions(mean, max, min, std dev) |
| 25 | 15.57 , 19 , 11 , 1.5764199947983404 |
| 50 | 21.23 , 24 , 18 , 1.2071039723238426 |
| 75 | 23.0 , 24 , 20 , 0.9695359714832658 |
| 100 | 23.74 , 24 , 22 , 0.46086874487211654 |
| 125 | 23.87 , 24 , 22 , 0.3648287269390939 |
| 150 | 23.98 , 24 , 23 , 0.13999999999999999 |

Color: “Red”, “Blue”

Text: “Red”, “Blue”

Derived Factor: True: “Red” “Red” is followed by “Blue”, “Blue”

False: otherwise

exclude(True-derived\_factor)

24 unique solutions

4!

18

4 trials 18 unique configs

1000 samples 100 times

390 sec 1000 samples

5-10 secs for 1000 samples in SAT solvers

Mean sdtdev

0 : 55.31 8.173244104510717

1 : 56.78 7.488145513564973

2 : 56.46 7.142729580390529

3 : 54.29 6.800616430764482

4 : 56.59 7.267048919609665

5 : 55.17 7.5746187022746225

6 : 55.31 7.412377036641709

7 : 55.45 6.744058365133211

8 : 55.76 6.960378487328738

9 : 55.74 7.574038592855241

10 : 55.03 7.649460270126411

11 : 54.81 7.412377036641709

12 : 55.42 7.115468992968699

13 : 55.69 7.637884047776107

14 : 55.64 6.823192771420107

15 : 54.75 6.546454048266177

16 : 56.79 7.873103427223446

17 : 55.01 7.060338790623566

original:

176.65679992198943 : 7.068615271297201

0 : 55.03 6.373009691288138

1 : 56.19 7.5259752212124

2 : 56.39 6.488365656225679

3 : 53.86 6.50721433363297

4 : 55.56 7.074381152218807

5 : 56.36 6.790546012767688

6 : 55.62 7.643284502084033

7 : 55.23 6.999069202184652

8 : 54.79 6.928633313717763

9 : 55.53 5.719813163579963

10 : 55.86 7.684458932866327

11 : 55.37 7.3891197655127305

12 : 57 7.3195462684347845

13 : 55.41 6.934170986695329

14 : 55.33 7.992490162014526

15 : 55.23 8.154883273972793

16 : 55.38 7.72609379004198

17 : 55.86 7.6515265526467715

improved

166.33532847215733 : 7.082613392071218

0 : 55.041666666666664 6.302742371659242

1 : 55.354166666666664 7.544923935533244

2 : 55.947916666666664 7.701496493903898

3 : 55.947916666666664 8.17617822401623

4 : 55.90625 7.982979426649914

5 : 55.5625 6.992947575198436

6 : 53.416666666666664 6.362168889809209

7 : 55.416666666666664 7.624982024712779

8 : 55.864583333333336 6.683238722697357

9 : 56.020833333333336 7.888671646790554

10 : 56.989583333333336 7.2395038964466805

11 : 56.479166666666664 7.246021291812736

12 : 55.15625 6.974129826874838

13 : 54.8125 7.145572792038143

14 : 55.15625 6.633770262432733

15 : 55.302083333333336 6.680875756754722

16 : 54.708333333333336 6.7307881241300285

17 : 56.916666666666664 7.15492714541108

100

18 calls ILP

cached:

3.7444076797749735 : 7.096771530261531

0 : 55.554455445544555 6.783030661179046

1 : 55.475247524752476 7.261672065586466

2 : 55.7029702970297 7.76085633735794

3 : 57.08910891089109 6.581943497024249

4 : 54.366336633663366 7.692493447871408

5 : 56.12871287128713 7.275525226863877

6 : 55 7.545859791965393

7 : 54.91089108910891 6.977247322405864

8 : 56.21782178217822 6.815576219801286

9 : 55.13861386138614 7.560462555915871

10 : 54.83168316831683 6.507025905789361

11 : 55.92079207920792 7.026639550050695

12 : 55.227722772277225 7.1398616066683145

13 : 55.56435643564357 7.382974795547061

14 : 55.56435643564357 7.671265660351176

15 : 55.53465346534654 7.0520413447960495

16 : 55.13861386138614 7.421630148384244

17 : 56.633663366336634 7.14384038494314