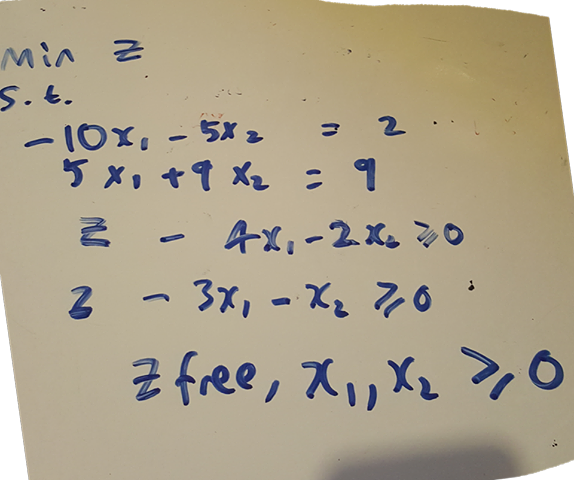
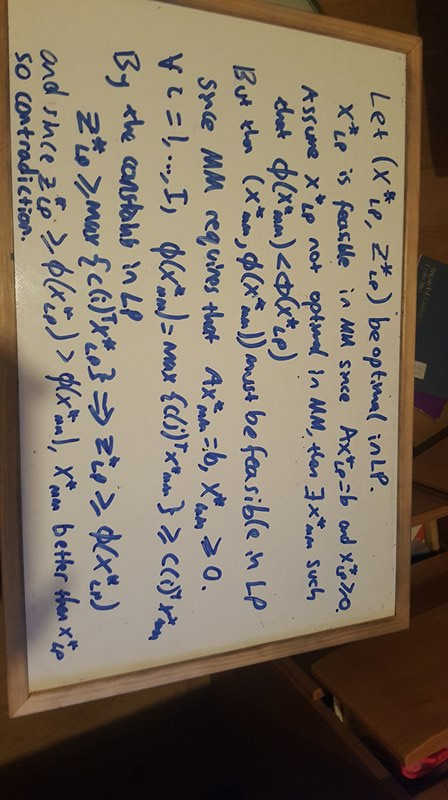
1 (a) Problem is in slides with 25 (5, 5), but solver can actually find 30 (0, 10); would be great if anyone knows why, which also answers the second part but naively I think that just wants a degeneracy comment.

(b) Tutorial problem

2a i



Ii



2bi) 0.75x3 + 0.25x4 - x5 + \xi\_1 = 0.75

(ii) rows 1 and 2 are <= constraints, observe the original LPs. Shadow prices for R1 is \beta\_3 (as R1 was standardised with slack x\_3), R2 \beta\_4.

3  
(a) RIP

Guess:   
x\_{XI} : how many applet X on i

s\_X : size of applet X

s\_Oi : how much of the object cache uses on i

d\_i : whether i is used

min sum (1024d\_i - (sum(s\_X\*x\_{Xi}) over all X) ) over all i

Sum (s\_X\*x\_{Xi} + s\_{Oi}) over all X <= 1024, for all i // do not exceed 1024MB cap

Sum (x\_{Ai}) over all i >= 5, for all i // 5A’s

Sum (x\_{Bi}) over all i >= 7, for all i

Sum (x\_{Ci}) over all i >= 1, for all i

Sum (s\_{Oi}) >= 128, for all i // Object cache larger than 128

Sum (s\_{Oi}) <= 512, for all i // object cache smaller than 512

d\_i <= sum (x\_{Xi}) over all x + s\_{Oi}, for all i // if i is used, set to 1, otherwise forced to 0

// below stuff are unsure stuff i formulate the logic, but cbf to translate into actual constraints

// s\_{Oi} can’t be on same chip as C, i.e. x\_{Ci} > 0 => s\_{Oi} = 0, equivalent to x\_{Ci} == 0 or s\_{Oi} = 0

// B can’t use any chip as C or its object cache, i.e. x\_{Ci} > 0 || s\_{Oi} > 0 => x\_{Bi} == 0

----------------------------------------------------

Alternative:

Let

pi - whether the memory bank **i** is on ∈ {0,1}

bi - memory used in bank **i** ≥ 0

axi - memory allocated by app of type **x** (or object cache if x=o) on **i** ≥ 0

ci - whether bank contains object cache ∈ {0,1}

nxi - the number of apps of type **x** running on **i** ∈ ℕ0

min ∑1024pi - bi

s.t.

Memory used does not exceed capacity:

bi ≤ 1024 ∀i = 1,...,8

Memory used is memory allocated by each app:

bi = aAi + aBi + aCi + aOi ∀i = 1,...,8

Set number of apps used for each type:

∑nAi = 5

∑nBi = 7

∑nCi = 1

Ensure each app allocates its whole mem to /single/ bank:

aAi = nAi \* 256 ∀i = 1,...,8

aBi = nBi \* 64 ∀i = 1,...,8

aCi = nCi \* 768 ∀i = 1,...,8

Set cache limits:

128 ≤ ∑aOi

∑aOi ≥ 512

Correctly(-ish) setting ci

aOi \* ci = aOi ∀i = 1,...,8

aOi ≥ ci // Need to somehow deal with situation where 0 < aOi < 1

Correctly setting pi, the power of the bank i

( nAi + nBi + nCi + ci ) \* pi = ( nAi + nBi + nCi + ci ) ∀i = 1,...,8

( nAi + nBi + nCi + ci ) ≥ pi ∀i = 1,...,8

Restrictions on type B apps:

nBi ≤ 0 + 7 (1 - nCi) ∀i = 1,...,8

nBi ≤ 0 + 7 (1 - ci) ∀i = 1,...,8

(b) (i) in case study (2^n)

(ii)in slides

4(a) (i)

// short hand, y\_k = k  
Min z’ = 4

2 - 3 + 4 >= 0

-1 + 3 + 4 >= 0

1 - 2 + 4 >= 0

1 + 2 + 3 = 1

1,2,3 >= 0; 4 free  
(ii)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1** | **2** | **3** |
| **1** | 0 | -1 | 1 |
| **2** | 1 | 0 | -1 |
| **3** | -1 | 1 | 0 |

Unique, but can have row/col permutations giving a different transpose (BONUS MEME: is the above a totally unimodular matrix?)

(iii) Nope, show working and state formula for Nash eqlbm in pure strats showing unequal should be sufficient

(iv) minimax theorem, strong duality giving that both problems if feasible and optimal have the same value, Nash eqlbm exists therefore.

4(b)

Because x\_1 is in {0, 1} choose branch P\_1 with x = 0 and second branch P\_2 with x = 1. Note that x\_2 does not need to be integer.

P\_1 : x = 0 giving -3 (0, 3)

P\_2 : x = 1 giving -3.5 (1, 1.5)

Hence answer is 3.5