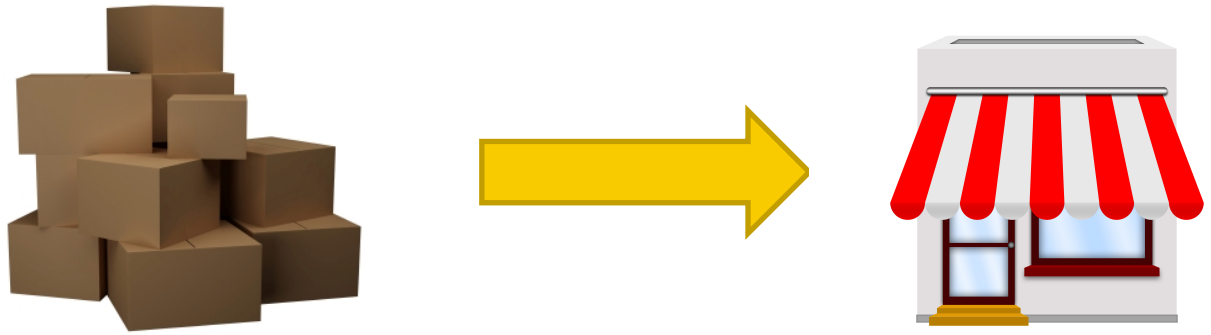


Optimisation de l'emballage pour envoi aux magasins
Optimization of the packing of boxes dispatched to retail stores

Cinquième atelier de résolution de problèmes de Montréal – Une activité CRM-Mprime --- Fifth Montréal Problem Solving Workshop – A CRM-Mprime Event

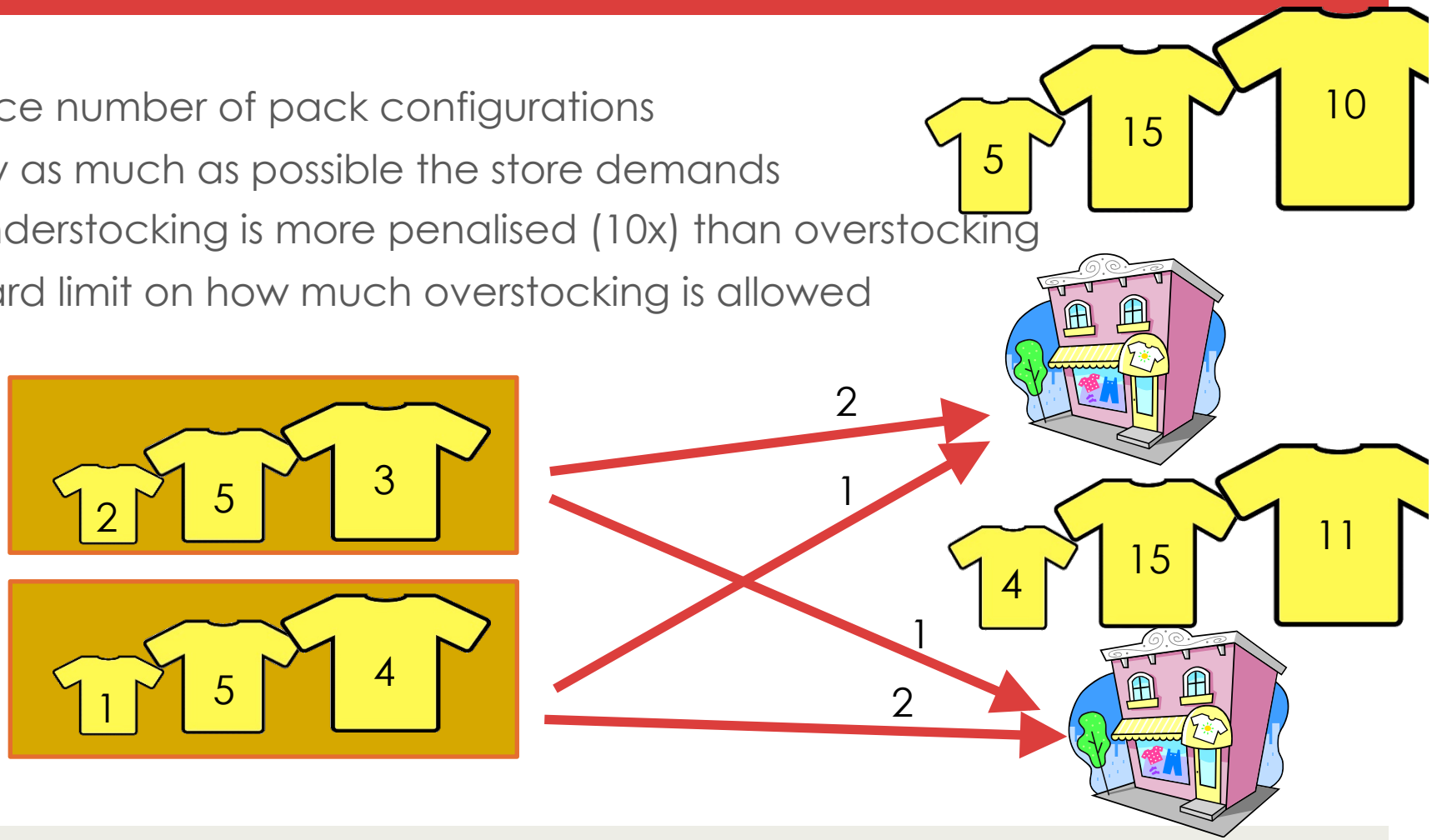
Sh. A. Alizadeh, Chris Breward, Mahsa Elahipanah, Éric Prescott-Gagon, Maxime Hoskins, Nahid Massoudi Renaud Masson, Gabrielle Gauthier Melançon, Jorge Mendoza, Christophe Meyer, Louis-Martin Rousseau, Akanksha Srivastava, Mohamed Sadoune, Winston Sweatman,

19 au 23 août 2013 - August 19-23, 2013

CRM, Université de Montréal

Prepack problem

- Reduce number of pack configurations
- Satisfy as much as possible the store demands
  - Understocking is more penalised (10x) than overstocking
  - Hard limit on how much overstocking is allowed



Mandate

- Identify and investigate a model of the problem
  - Fast Heuristics Approach
  - Exact Formulation
- Look at the advantages of building « rainbow » pack

Divide and Conquer

- Our team was quite large (10+) so we investigated different approaches in parallel.
  - Constraint Programming
  - MIP – Compact Formulation
  - A (very)Hybrid Metaheuristic
  - HumanSearch

Constraint Programming

- **Objective: Minimize over and under Stock**

$$\min \sum_{i,j,s} (10 * under(i,j,s) + over(i,j,s))$$

- **Demand Constraint (non linear)**

$$\sum_{b} fill(b,i,j) * \sum_{s} send(b,s) = \sum_{s} (dem(i,j,s) + over(i,j,s) - under(i,j,s))$$

- **Overstock should be less than demand**

$$over(i,j,s) \leq \sum_{j} Demand(i,j,s) \bigwedge over(i,j,s) \leq overstock(i,j)$$

- **We fix the number of boxes to given Value**

Constraint Programming

- Limit the number of item that go in a box to demand

$$fill(b,i,j) \leq \max_{s} dem(i,j,s)$$

- Ship enough Boxes to meet demand

$$send(b,s) \leq \sum_{i,j} \lceil dem(i,j,s)/4 \rceil$$

- Lower Bound (based on Odd/Even property)

$$Obj \geq \sum_{s} \mod \left(\sum_{i,j} dem(i,j,s), 2\right)$$

- What you Ship is What you Get

$$\sum_{b} capa(b) * send(b,s) = \sum_{i,j} dem(i,j,s) + over(i,j,s) - under(i,j,s)$$

MINLP – Compact Formulation

$$\min \quad \sum_{b=1}^{B} \sum_{c \in C} t_{bc}$$

$$\min \quad \alpha \sum_{s \in S} \sum_{i \in I} \ell_{is} + \beta \sum_{s \in S} \sum_{i \in I} e_{is}$$

$$z_{is} = \sum_{b=1}^{B} \sum_{\ell=0}^{\lfloor \log \overline{X}_{bs} \rfloor} 2^{\ell} w_{bis\ell} \qquad i \in I; \quad s \in S$$

$$z_{is} - e_{is} + \ell_{is} = \text{DEMAND}_{is} \qquad i \in I; \quad s \in S$$

$$z_{is} = \sum_{b=1}^{B} x_{bs} y_{bi} \qquad i \in I; \quad s \in S$$

$$\sum_{i \in I} y_{bi} = \sum_{c \in C} c t_{bc} \qquad b = 1, \ldots, B$$

$$\sum_{c \in C} t_{bc} \leq 1 \qquad b = 1, \ldots, B$$

$$e_{is} \leq \text{MAXOVERSTOCK}_{is} \qquad i \in I; \quad s \in S$$

$$t_{bc} \in \{0; 1\} \qquad b = 1, \ldots, B; \quad c \in C$$

$$x_{bs} \text{ integer and } \geq 0 \qquad b = 1, \ldots, B; \quad s \in S$$

$$y_{bi} \text{ integer and } \geq 0 \qquad b = 1, \ldots, B; \quad i \in I$$

$$e_{is}, \ell_{is} \geq 0 \qquad i \in I; \quad s \in S$$

Linearizing to a MIP

- First make the x variable binairies

$$x_{bs} = \sum_{\ell=0}^{\lfloor \log \overline{X}_{bs} \rfloor} 2^{\ell} u_{bs\ell} \qquad b = 1, \ldots, B; \quad s \in S$$

$$u_{bs\ell} \in \{0; 1\} \qquad b = 1, \ldots, B; \quad s \in S; \quad \ell = 0, \ldots, \lfloor \log \overline{X}_{bs} \rfloor$$

- Then Linearize the model

$$w_{bis\ell} \leq \overline{Y}_{bi} u_{bs\ell} \qquad b = 1, \ldots, B; \quad i \in I; \quad s \in S; \quad \ell = 0, \ldots, \lfloor \log \overline{X}_{bs} \rfloor$$

$$w_{bis\ell} \leq y_{bi} \qquad b = 1, \ldots, B; \quad i \in I; \quad s \in S; \quad \ell = 0, \ldots, \lfloor \log \overline{X}_{bs} \rfloor$$

$$w_{bis\ell} \geq 0 \qquad b = 1, \ldots, B; \quad i \in I; \quad s \in S; \quad \ell = 0, \ldots, \lfloor \log \overline{X}_{bs} \rfloor$$

$$w_{bis\ell} \geq y_{bi} - \overline{Y}_{bi}(1 - u_{bs\ell}) \qquad b = 1, \ldots, B; \quad i \in I; \quad s \in S; \quad \ell = 0, \ldots, \lfloor \log \overline{X}_{bs} \rfloor$$

Improving the formulation

We can further improve this model by adding tighter upper bounds…

Computed via an additional MIP model

v_{bs}: binary variable equal to 1 if any box type b is sent to store s
u_{bs} number of box type b sent to store s

$$\min w_1 \sum_i \sum_s \left(p_1 y_{is}^1 + p_2 y_{is}^2\right) + w_2 \sum_b \sum_s u_{bs},$$

s.t.:

$$\sum_b u_{bs} c_b + \sum_i y_{is}^1 = \sum_i d_{is} + \sum_i y_{is}^2, \ \forall s \in S$$

$$y_{is}^2 \leq o_i, \ \ \forall i \in I, s \in S$$

$$M v_{bs} \geq u_{bs}, \ \ \forall b \in B, s \in S$$

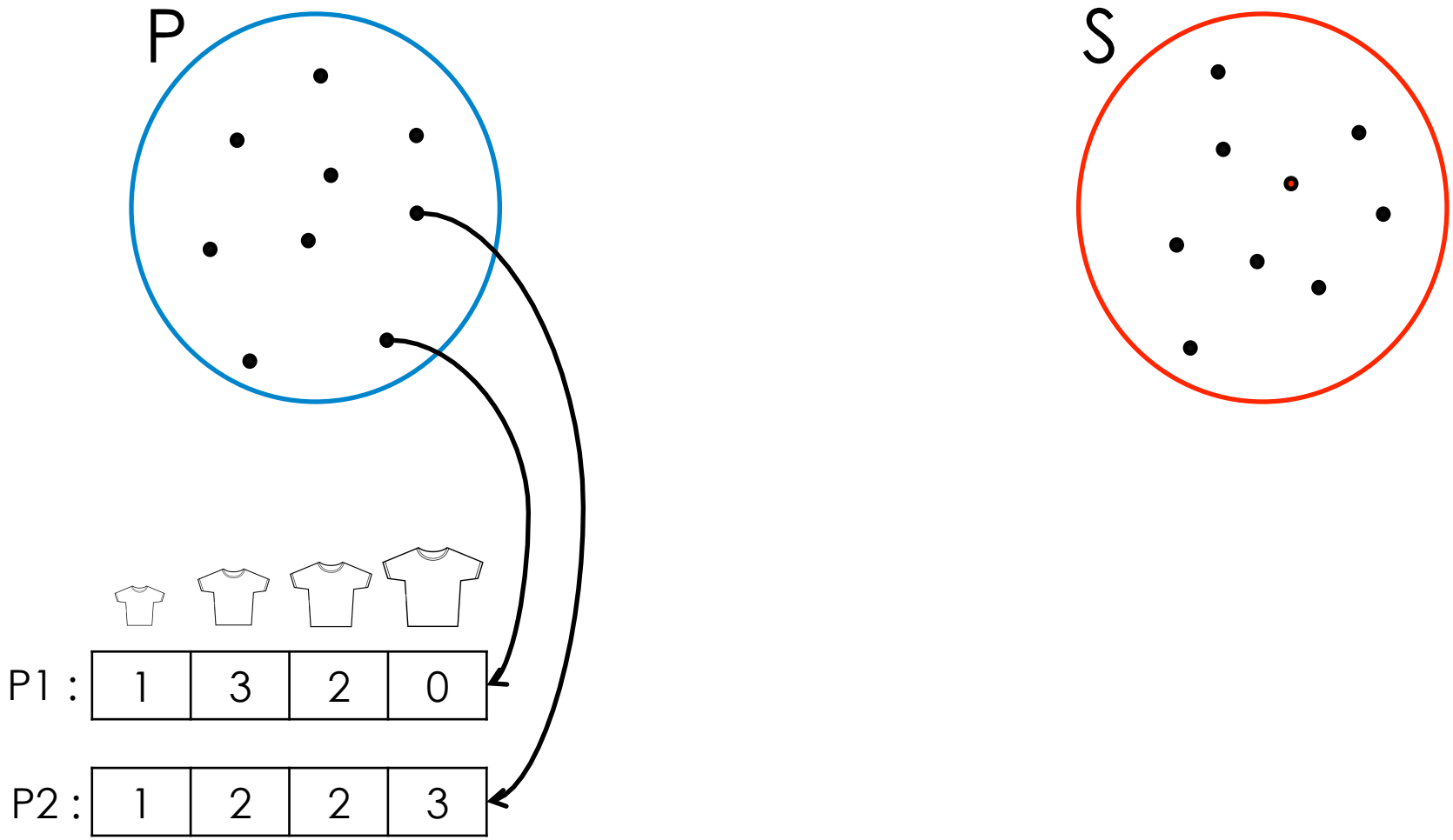
We get the following as the output of this model:
U_{bs}: upper bounds for total number of box type b sent to store s.

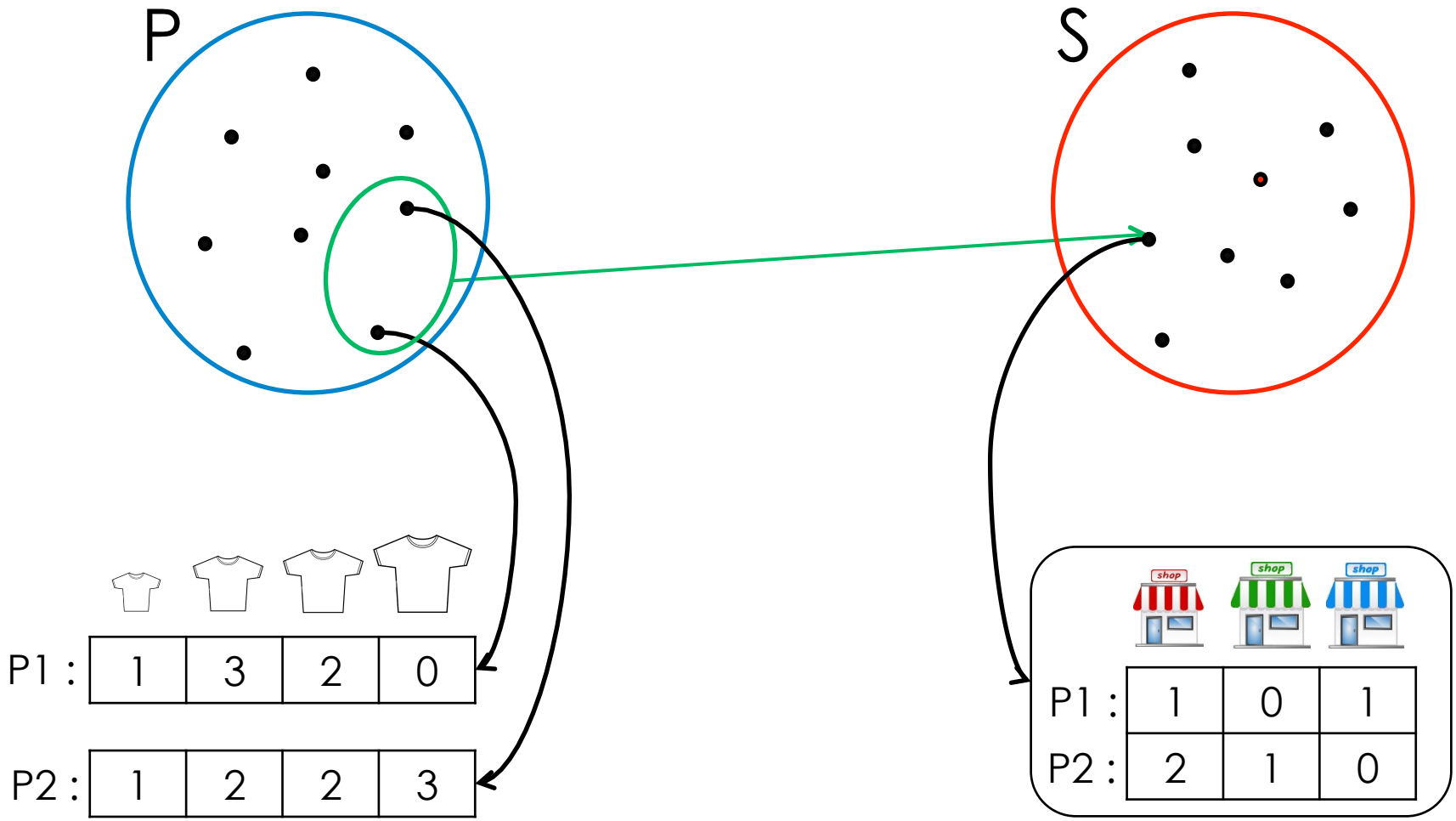
$$U_{cs} = U_{bs} \times \left[\begin{array}{cccc} I_{|B|} & \ldots & I_{|B|} \end{array} \right]_{|B| \times |M|}$$

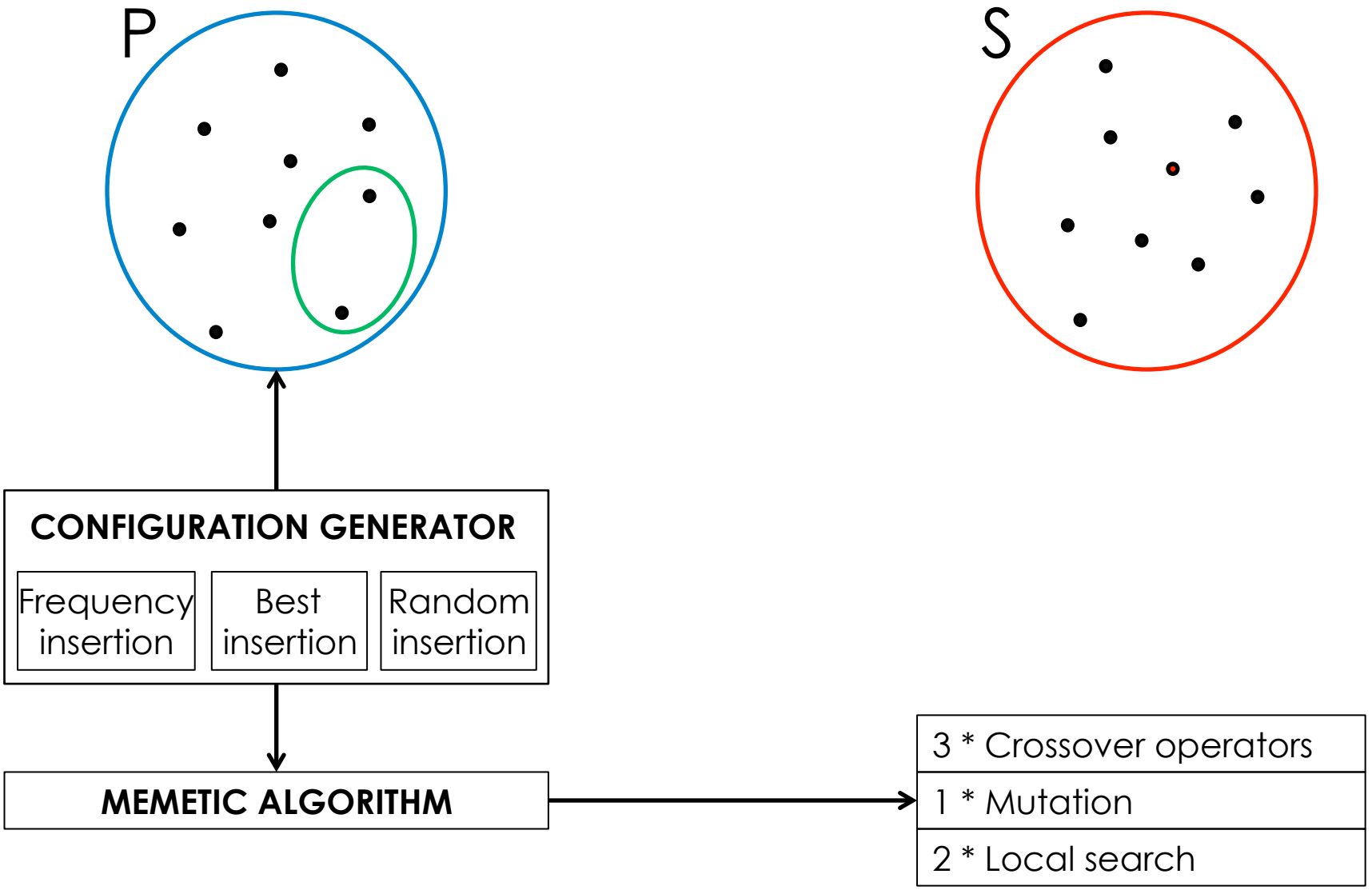
$$J_{cs} = \{0, ..., \log_2 U_{cs}\}, \forall c \in C, \forall s \in S$$

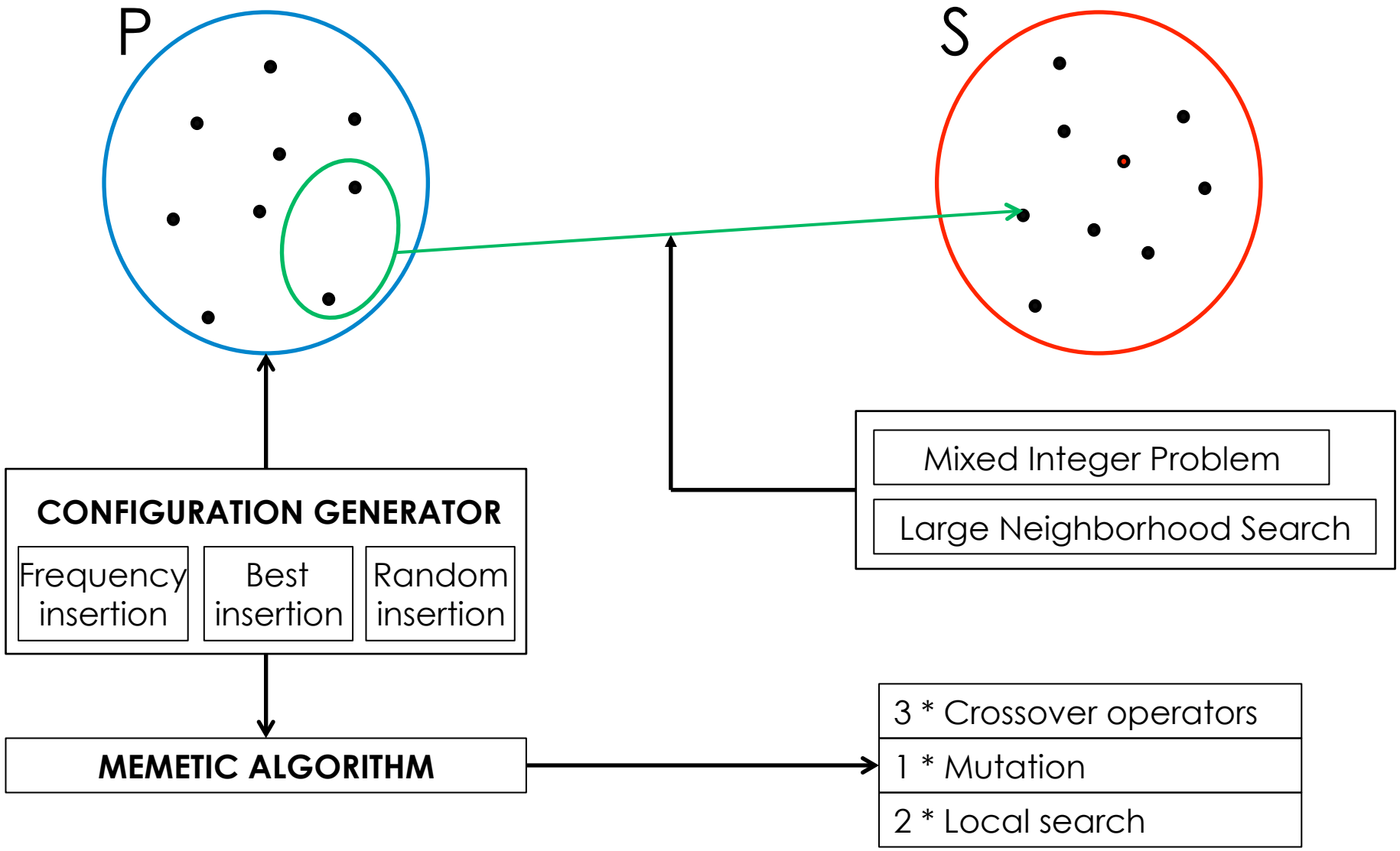
Muesliristic

- A Good and Healty Combination of
  - Construction Heuristics
  - Memetic Algorithm
  - Integer Program
  - Large Neighborhood Search


P
S
P1 :
1
3
2
0
P2 :
1
2
2
3


P
S
P1 :
1
3
2
0
P2 :
1
2
2
3
shop
shop
shop
P1 :
1
0
1
P2 :
2
1
0


P
S
CONFIGURATION GENERATOR
Frequency insertion
Best insertion
Random insertion
MEMETIC ALGORITHM
3 * Crossover operators
1 * Mutation
2 * Local search


P
S
Mixed Integer Problem
Large Neighborhood Search
CONFIGURATION GENERATOR
Frequency insertion
Best insertion
Random insertion
MEMETIC ALGORITHM
3 * Crossover operators
1 * Mutation
2 * Local search

Mixed Integer Problem

$$\min \quad \sum_{c \in C} \sum_{s \in S} \alpha u_{is} + \beta o_{is} \qquad (1)$$

$$\text{s.t.} \quad \sum_{c \in C} a_{ic} x_{cs} = d_{is} + o_{is} - u_{is} \qquad (2)$$

Large Neighborhood Search

$$\min \quad \sum_{c \in C} \sum_{s \in S} \alpha u_{is} + \beta o_{is} \qquad (3)$$

$$\text{s.t.} \quad \sum_{c \in C} a_{ic} x_{cs} = d_{is} + o_{is} - u_{is} \qquad (4)$$

$$\sum_{s \in S} x_{cs} \leq M y_c \qquad \forall c \in C \qquad (5)$$

$$\sum_{c \in C} y_c = nbConfig \qquad (6)$$

The power of the humain Brain

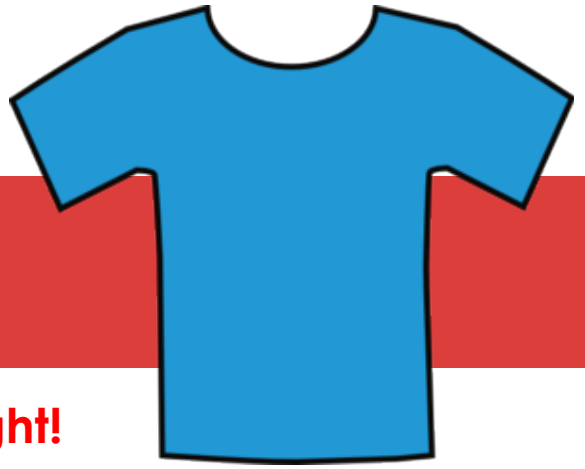
CW's Secret Sauce

- Identify types of store demand vector (ST_i)
- Find common vector (C) required by all stores
- Calculate residual vector $R_i=ST_i-C$
- Initially do one colour at a time (demand is colour structured)
- **Find spanning** set $\{S_j\}$ of R_i
- **Divide** common vector (C) into parts (C1, C2 etc)
- **Make** boxes Bi= $\{S_j\}$+ C1 to form box vectors with 4,6,8,10 T-shirts
- **Take care** with odd/even T-shirt demands, and in cases where 1 store has zero demand in a colour
- **Combine** C2s between colours if possible

Blue T-Shirts only

- Consider blue T-shirts as a simple example.
  - 4 different store types with vectors:
    ST1=(1,2,2,5,3,1), ST2=(1,2,3,3,4,1), ST3=(1,2,4,3,3,1), ST4=(1,2,3,4,3,1)
- Each store requires 14 blue T-shirts;
  - Common vector C=(1,2,2,3,3,1)
- Residual vectors are
  - R1= (0,0,0,2,0,0), R2=(0,0,1,0,1,0), R3=(0,0,2,0,0,0), and R4=(0,0,1,1,0,0)
- The spanning set is
  - S1=(0,0,0,1,0,0), S2=(0,0,1,0,1,0), S3=(0,0,1,0,0,0)
- R1=2S1, R2=S2, R3=2S3 and R4=S1+S2
- Size of C is 12 (bigger than max box size)

Blue T-Shirts only

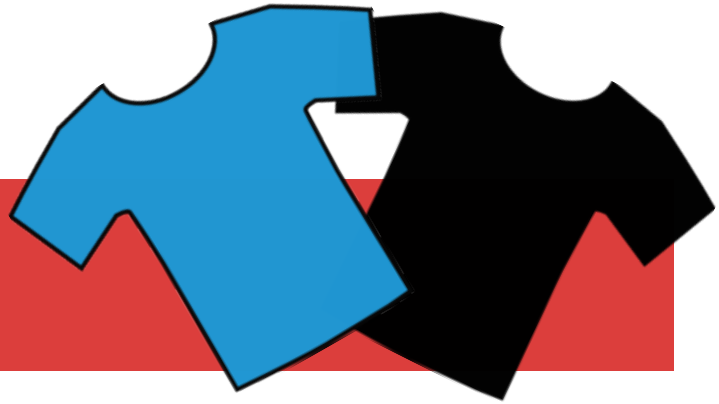


The next step relies on considerable hindsight!

- Split C=2C1+C2, where C1 is active part and C2 is passive part (C1 must be odd and must have length 3). Pick C1=(0,1,1,1,0,0) and C2=(1,0,0,1,3,1)
  - B1=C1+S1=(0,1,1,2,0,0)
  - B2=2C1+S2=(0,2,3,2,1,0)
  - B3=C1+S3=(0,1,2,1,0,0)
  - B4=C2=(1,0,0,1,3,1)
- Make boxes by adding the spanning vectors to multiples of the active box
- The supply is then
  - ST1= 2B1+B4
  - ST2=B2+B4
  - ST3=2B3+B4
  - ST4=B1+B3+B4

All stores get precisely the number of T-shirts they need, using 4 box types

Blue and Black



If we look at boxing up the blue and black T-shirts, two solutions are:

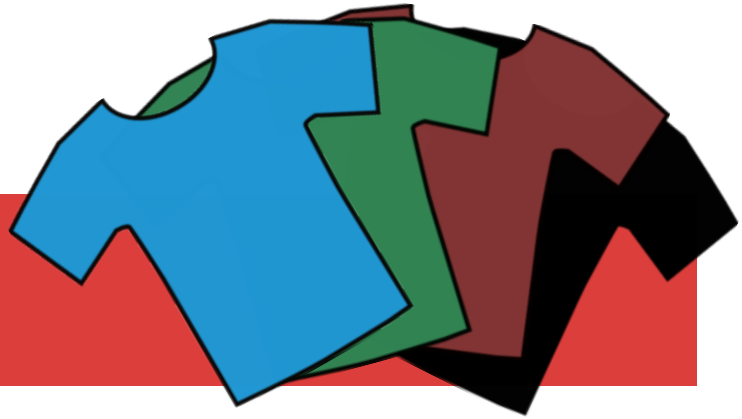
B1=(1,0,2,2,4,1,0,0,0,0,0,0)
B2=(1,0,4,2,2,1,0,0,0,0,0,0)
B3=(1,0,2,4,2,1,0,0,0,0,0,0)
B4=(0,0,0,0,0,0,0,1,1,2,0,0)
B5=(0,0,0,0,0,0,0,2,3,2,1,0)
B6=(0,0,0,0,0,0,0,1,2,1,0,0)
B7=(0,2,0,2,0,0,1,0,0,1,3,1)

B1=(0,1,3,1,1,0,0,1,1,1,1,0)
B2=(0,1,1,3,1,0,0,1,1,1,1,0)
B3=(0,1,1,1,3,0,0,1,1,1,1,0)
B4=(0,1,1,1,1,0,0,1,3,1,1,0)
B5=(0,1,1,1,1,0,0,1,1,3,1,0)
B6=(0,1,1,1,1,0,0,1,2,1,2,0)
B7=(1,0,0,1,0,1,1,0,1,1,1,1)

- In LHS solution, one box is multicoloured - the passive box that goes to everyone
- In the RHS solution, all boxes are multicoloured

Note that because all stores require 13 black T-shirts, all stores the all get an extra T-shirt (black in LHS solution, blue in RHS)

All colours



With all four colours:

- Green T-Shirts decouple from the problem (takes 3 box types)
- Solution involves 14 box types
  - 10 boxes are monocolour, 4 boxes are multicolour
  - Multicolour boxes reduce the need for oversupply to stores which need odd number of red T-shirts as well as odd number of black T-shirts
  - 10 stores get exactly the right number; 20 stores get one extra black T-shirt; 1 store (the one with no red T-shirts) gets one fewer black T-Shirt
- No way to improve this solution with 14 boxes

Fewer boxes than 14:

- Using fewer boxes than 14 necessarily increases the over/under supply.
- For 13, best strategy is to take the overlap between two green boxes (taking care to make box size even).
- Supply 32 fewer T-shirts overall.
- Can reduce to one green type:
  - supply 50 fewer T-shirts overall (2 per store!)

And the Winner is …

- Constraint Programming (35 lines of AIMMS)
  - Can solve monochrome problem instantly
  - Slow on rainbow-pack… solution with understock
- Linearized MIP (XX lines of C++)
  - Can solve monochrome problem easily
  - Similar to CP on rainbow-pack
- Muesliristic (4845 lines of JAVA)
  - Tackles the global problem, find a solution with 14 boxes 59 under stock and 199 over stock. (out 47000 explored box-types)
- CW-brains (?? Kcal, ? Coffees, ? Excell Tabs)
  - Find a solution to the global problem with 14 box, no under cover, 74 over cover

Many thanks to

- Workshop organizers and sponsors
- JDA / Éric Prescott Gagon for his help and support
- All the member for their enthusiasm and collaborative spirit.