PA184 - Heuristic Search Methods

Lecture 6 – Constraint Handling Techniques

- General Considerations
- Tackling Constraints
  - Rejecting Infeasible Solutions
  - Repairing Infeasible Solutions
  - Penalising Infeasible Solutions
  - Enforcing Feasible Solutions
  - Treating Constraints As Objectives

Learning outcomes:

- Identify the key issues to be considered when dealing with constraints in heuristic search
- Understand the various methods to handle constraints within heuristic search methods

General Considerations

In optimisation problems the goal is to find the best feasible solution.

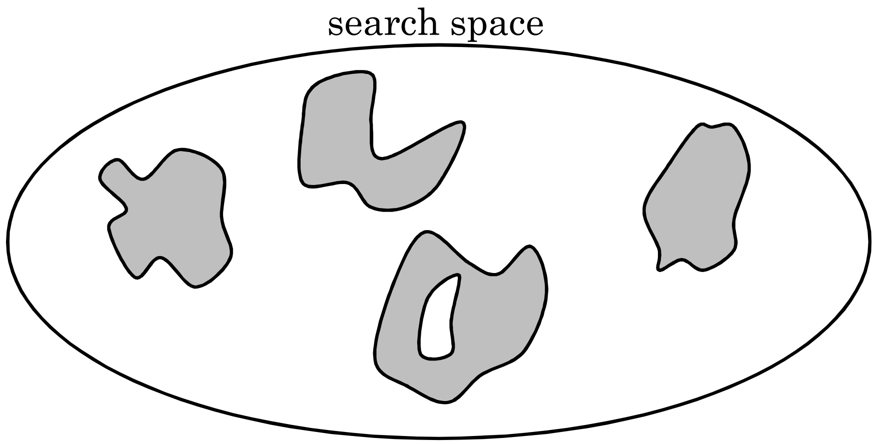
The key question when dealing with constraints is:

Should infeasible solutions be considered during the search?

or

Should infeasible solutions be rejected straight away?

The evaluation function plays a crucial role in ranking solutions both feasible and infeasible.

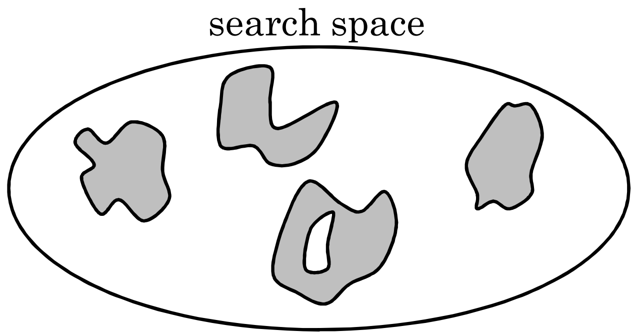


The search space usually consists of disjoint subsets of feasible (F) and infeasible (I) solutions, although there is no guarantee that these subsets are connected.

Issues When Dealing with Infeasible Solutions

- Evaluate quality of feasible and infeasible solutions
- Discriminate between feasible and infeasible solutions
- Decide if infeasibility should be eliminated
- Decide if infeasibility should be repaired
- Decide if infeasibility should be penalised
- Initiate the search with feasible solutions, infeasible ones or both
- Maintain feasibility with specialised representations, moves and operators
- Design a decoder to translate an encoding into a feasible solution
- Focus the search on specific areas, e.g. in the boundaries

A constraint handling technique is required to help the effective and efficient exploration of the search space.



Examples of Constrained COPs

Multiple-knapsack problem

n items each of size $s_i > 0$

m knapsacks each of capacity $b_j > 0$

profit $p_{ij} > 0$ for assigning item i to knapsack j

Maximise $\quad Z = \sum_{j=1}^{m} \sum_{i=1}^{n} p_{ij} x_{ij}$

subject to $\quad \sum_{i=1}^{n} s_i x_{ij} \le b_j \quad$ for $j = 1,2,\ldots m \quad (1)$

$\sum_{j=1}^{m} x_{ij} \le 1 \quad$ for $i = 1,2,\ldots n \quad (2)$

$x_{ij} = \{0,1\}$

Bin-packing problem

n items each of size $s_i > 0$

m bins each of equal capacity $b > 0$

Minimise $\quad Z = \sum_{j=1}^{m} y_j$

subject to $\quad \sum_{i=1}^{n} s_i x_{ij} \le y_j b \quad$ for $j = 1,2,\ldots m \quad (1)$

$\sum_{j=1}^{m} x_{ij} = 1 \quad$ for $i = 1,2,\ldots n \quad (2)$

$x_{ij} = \{0,1\}$ and $y_j = \{0,1\}$

Generalised assignment problem

n tasks and m workers

worker j takes $t_{ij} > 0$ time to complete task i

worker j has $T_j > 0$ time available

assigning task i to worker j has a cost $c_{ij} > 0$

Minimise $\quad Z = \sum_{j=1}^{m} \sum_{i=1}^{n} c_{ij} x_{ij}$

subject to $\quad \sum_{j=1}^{m} x_{ij} = 1 \quad$ for $i = 1,2,\ldots n \quad (1)$

$\sum_{i=1}^{n} t_{ij} x_{ij} \leq T_j \quad$ for $j = 1,2,\ldots m \quad (2)$

$x_{ij} = \{0,1\}$ and $c_{ij}, t_{ij}, T_j > 0$

School timetabling problem

m classes, n teachers and p timeslots

$R_{m \times n}$ is a matrix where

r_{ij} is number of times teacher j must meet class i

Find $\quad x_{ijk} \quad$ for $i = 1\ldots m$; $j = 1\ldots n$; $k = 1\ldots p$

s.t. $\quad \sum_{k=1}^{p} x_{ijk} = r_{ij} \quad$ for $i = 1\ldots m$; $j = 1\ldots n \quad (1)$

$\sum_{j=1}^{n} x_{ijk} \leq 1 \quad$ for $i = 1\ldots m$; $k = 1\ldots p \quad (2)$

$\sum_{i=1}^{m} x_{ijk} \leq 1 \quad$ for $j = 1\ldots n$; $k = 1\ldots p \quad (3)$

$x_{ijk} = \{0,1\}$

Designing the Evaluation Function

For some problems like BIN PACKING and SCHOOL TT, designing an appropriate evaluation function to provide a good search landscape is not trivial.

To compare solutions an ordering relation might be required instead of a detailed evaluation function.

A common way to evaluate infeasibility is to add a component to the evaluation function:

$$\text{fitness\_function}(x) = \text{evaluation}_{\text{feasible}}(x) \pm \text{evaluation}_{\text{infeasible}}(x)$$

The $\text{evaluation}_{\text{infeasible}}(x)$ component represents a penalty or cost and it is also difficult to design it correctly to facilitate the search.

Key issue: to compare a feasible solution x_{fea} and an infeasible solution x_{inf} when x_{inf} is perhaps 'next' to the optimal solution x^*.

Tackling Constraints

Rejecting Infeasible Solutions

This is a simple strategy that has serious limitations particularly on those problems where generating feasible solutions is considerably more difficult that generating infeasible ones.

A heuristic search algorithm might need to cross between the feasible and infeasible regions of the search space.

In local search, assessing the feasibility of neighbour solutions might be helpful as it could help to obtain an idea of the fitness landscape topology.

Important issue: assess $\frac{|S_F|}{|S|}$ where,

S is the search space under consideration

S_F is the feasible portion of S

Enforcing Feasible Solutions

The use of specialised representations, decoders, neighbourhood moves and reproduction operators is a reasonable way to avoid generating infeasible solutions.

The characteristics of a good decoder are as follows:

- For each feasible solution there is an encoded one
- Each encoded solution corresponds to a feasible solution
- All feasible solutions should be represented by the same number of encoded ones
- The decoding procedure is computationally fast
- Small changes to the encoded solution represent small changes to the feasible solution

For example, a decoder can be used to convert a binary string into a solution to the BIN PACKING problem.

Repairing Infeasible Solutions

This is a popular strategy to handle constraints in heuristic search where an infeasible solution x_{inf} is transformed into a repaired feasible solution x_{rep}.

Then, x_{rep} can replace x_{inf} or it might be that x_{inf} remains but it is evaluated as x_{rep}.

It is possible that in a population-based heuristic only a number of infeasible solutions are repaired in order to maintain diversity.

The major weakness of the repairing strategy is that is very problem dependant.

For example, in the BIN PACKING problem a common repair strategy would be simply to remove items from the overloaded bin to another open or new bin.

Penalising Infeasible Solutions

The key issue is to establish the relationship between the infeasible solution and the feasible part of the search space in order to assess the value of the components in the infeasible solution.

$$x_{inf} \Leftrightarrow S_{feasible} \begin{cases} \bullet \text{ fixed penalty for infeasibility} \\ \bullet \text{ penalty proportional to degree of infeasibility} \\ \bullet \text{ penalty proportional to cost of repairing} \end{cases}$$

Penalties applied can be fixed or varied as the search progresses.

When tuning penalty functions: discrimination vs. exploration

For example, for the BIN PACKING problem a way to penalise infeasible solutions can be to add a weighted penalty due to the amount of exceeded capacity.

Some guidelines when designing constraint penalties:

- Penalties should that indicate 'distance from feasibility' in order to guide the search more effectively.
- If the problem has few constraints, just counting the number of constraint violations is generally not effective.
- Good penalty functions (static or dynamic) can be constructed by estimating the completion cost, i.e. the increase in the cost function (considering minimisation) incurred when transforming an infeasible solution into a feasible one.
- There is a compromise for the accuracy of the penalty function. It should be able to differentiate among infeasible solutions and not be too expensive to compute.
- A compromise for the weight value associated to the violation of each constraint should be found. Too small weights might produce final infeasible solutions. Too large weights might provoke convergence to suboptimal feasible solutions.

Treating Constraints As Objectives

Another strategy is to set an evaluation function for each constraint (or group of constraints) in addition to the evaluation$_{\text{fitness}}$(x) function and then to tackle with a multi-objective approach.

This approach is particularly useful when just finding feasible solutions is very difficult as this provides a smoother fitness landscape for the heuristic search to operate.

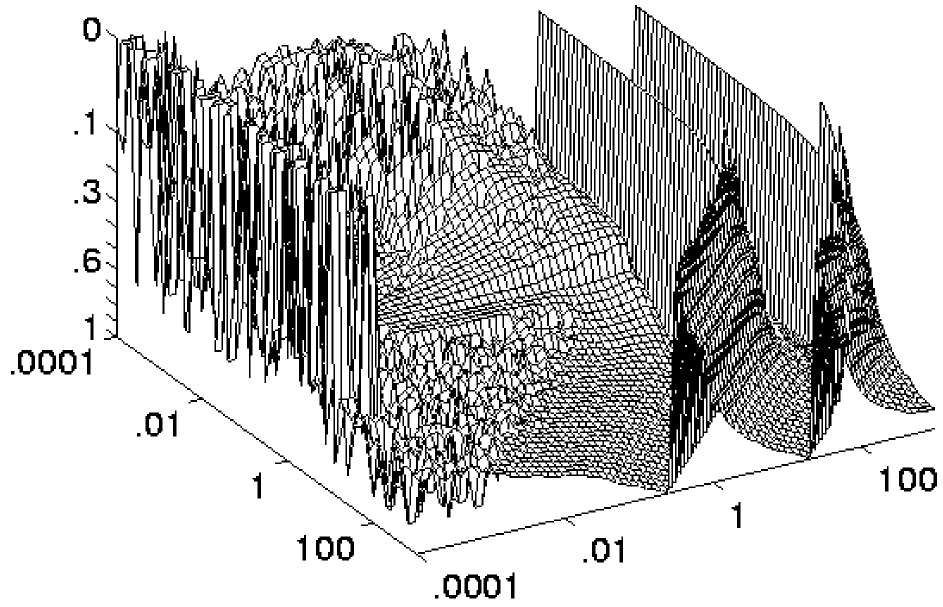
It is important to choose appropriate scales for the different objectives (maybe normalise them) so that the search is not biased.

Having several objectives then allows the application of MOO techniques like: weighted aggregating functions, Pareto optimisation, goal programming, etc.

What Makes a Constrained Problem Difficult?

One or several of the following aspects can make a constrained optimisation problem difficult to tackle with heuristic search:

- Size of the problem in terms of the number of decision variables
- Number of constraints (linear, non-linear, equality, inequality, etc.)
- The size of the search space and ratio of the feasible part to the whole.
- The fitness landscape induced by the evaluation function
- Number of local and global optima
- Etc.



Shelf Space Allocation Problem

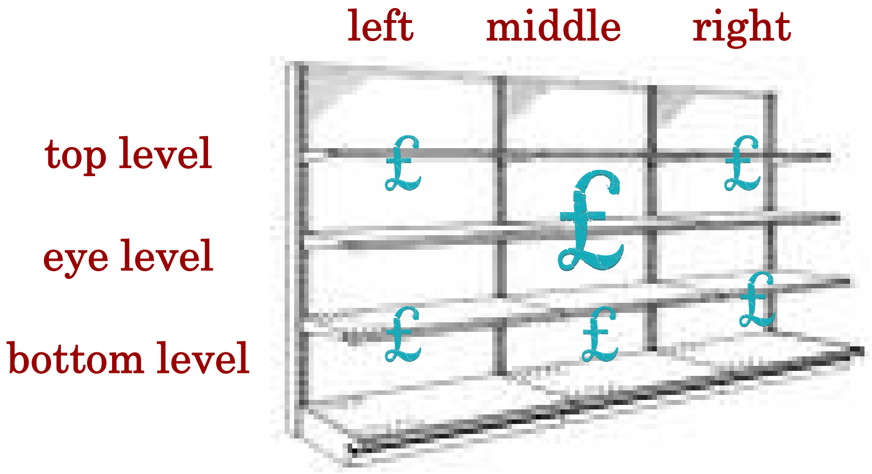
A **block** is a group of shelves

A **shelf** is a horizontal unit of space and can be: top level, eye-level or bottom level

A shelf is (virtually) divided into **3 parts**: left, middle and right

Priorities are assigned to shelves and parts by the manager

Eye-level shelves and centre parts assigned the **highest profitability**



What Data is Given?

For shelves and parts:

- Number
- Length
- Priority

For products:

- Length of a facing
- Maximum facings required
- Minimum facings required
- Number of available units
- Profit per unit
- Profitability according to location

Additional considerations:

- Product selection stage is not required
- Height and depth of a product unit are ignored
- Generally, all units of the same product are allocated in contiguous space
- No elaborate product categorisation is used

A Simple Formulation

Maximise $$Z = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p} \varphi_{ijk} x_{ijk} \qquad (1)$$

subject to $$\sum_{i=1}^{n} a_i x_{ijk} \le T_{jk} \quad \text{for} \quad j = 1\ldots m, k = 1\ldots p \qquad (2)$$

$$\sum_{j=1}^{m}\sum_{k=1}^{p} y_{ijk} = 1 \quad \text{for} \quad i = 1\ldots n \qquad (3)$$

$$L_i \le \sum_{j=1}^{m}\sum_{k=1}^{p} x_{ijk} \le U_i \quad \text{for} \quad i = 1\ldots n \qquad (4)$$

$$y_{ijk} \le x_{ijk} \le U_i y_{ijk} \quad \text{and} \quad x_{ijk} \in Z^+ \qquad (5)$$

n : products to allocate

m : shelves availabe

p : shelf parts ($p = 3$ here)

a_i : length of facing

L_i : minimum facings

U_i : maximum facings

x_{ijk} : facings i in part j of shelf k

y_{ijk} : facings i allocated to part j of shelf k?

ρ_i : profit per unit

σ_j : priority of shelf j

ω_k : priority of shelf part k

$\varphi_{ijk} = \rho_i \sigma_j \omega_k$: estimated profitability

A Tailored Heuristic Method

1. Preparatory Phase

Is there enough space to allocate all products?

$$\sum_{i=1}^{n} L_i a_i \leq \sum_{j=1}^{m} \sum_{k=1}^{p} T_{jk}$$

2. Allocation Phase

Produce an initial arrangement

Product profit

Product size ρ_i / a_i

Random choice a_i

Existing arrangement

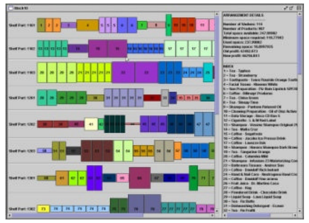
3. Adjustment Phase

Iterative changes to improve overall profit

Swap
AdjustIn
MultiShift
AdjustInter
RemoveLeastProfitable

4. Termination Phase

Compute quality of solution and display planogram

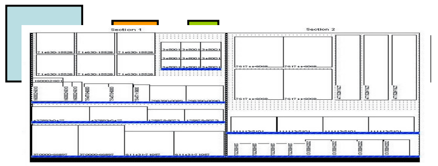


2. Allocation Phase

Prioritise

P^S : sorted products, ↓ order ρ_i/a_i or ↑ order a_i or random

S^S : sorted shelf parts, ↓ order of σ_j and ω_k

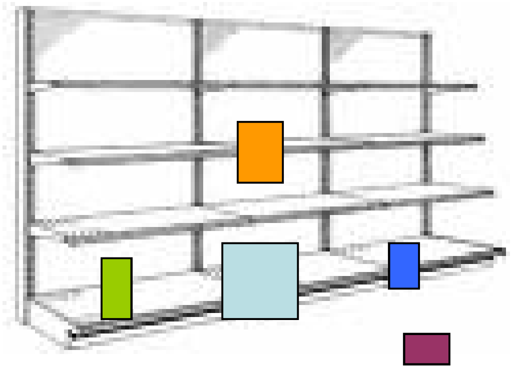


Allocate Minimum

Given: P^S and S^S produce arrangement Λ

Take next product i and next shelf part jk

Allocate L_i units to shelf part jk

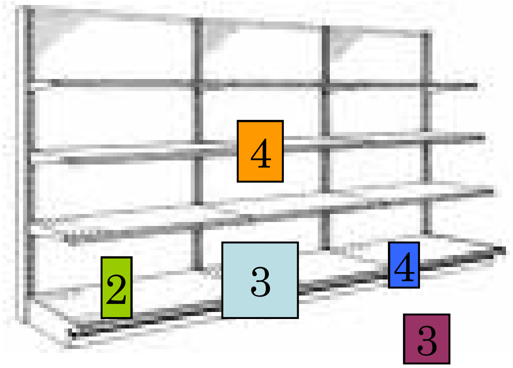


Allocate More

Given: P^S and arrangement Λ, improve Λ

Take next product i

Increment x_{ijk} by 1 ensuring feasibility (U_i and T_{jk})



3. Adjustment Phase

Swap

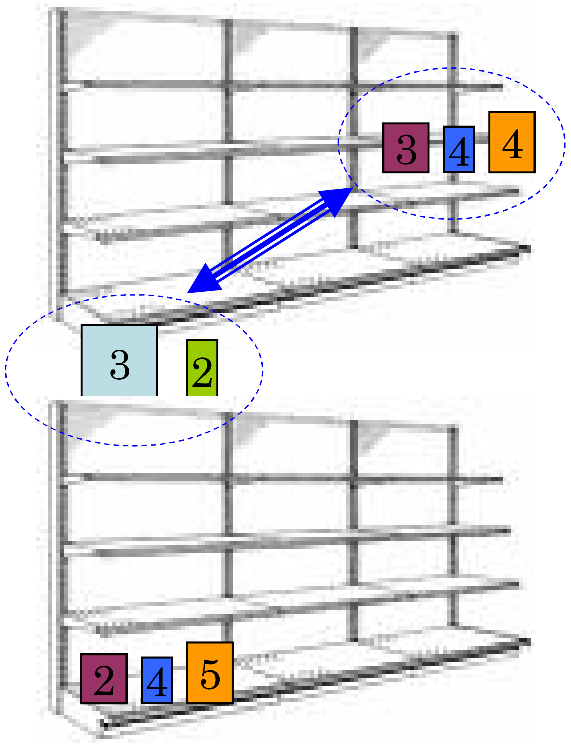
All facings between products i_1 and i_2 located in different shelf parts j_1k_1 and j_2k_2

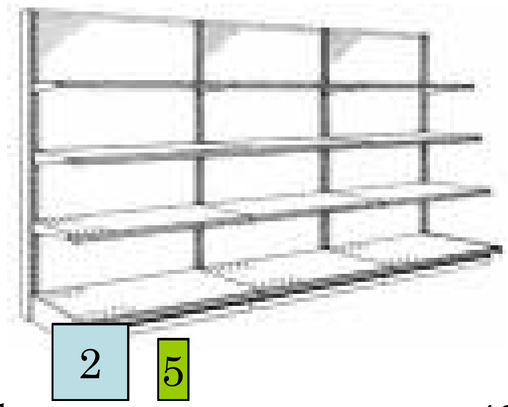
AdjustIn

For products i_1 and i_2 both in shelf part jk make $x_{i1jk} = x_{i1jk}+1$ and $x_{i2jk} = x_{i2jk} - 1$

MultiShift

For products i_1 and i_2 both in shelf part jk make $x_{i1jk} = x_{i1jk}+\alpha_1$ and $x_{i2jk} = x_{i2jk} - \alpha_2$

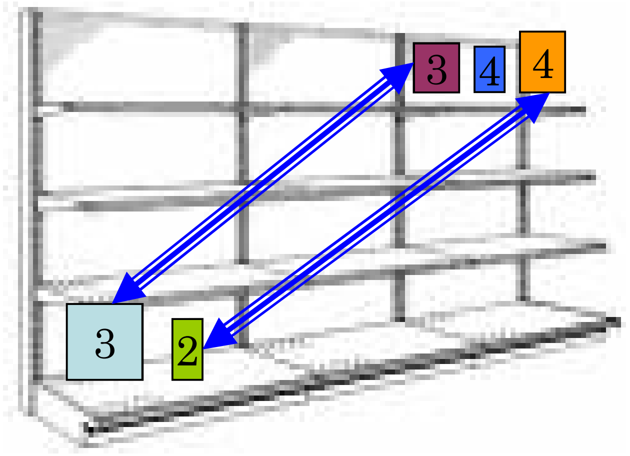




AdjustInter

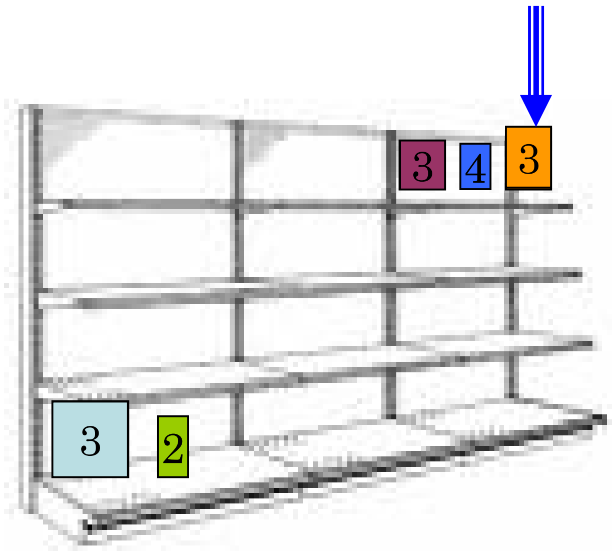
For products i_1 and i_2 both in shelf part j_1k_1 and products i_3 and i_4 both in shelf part j_2k_2 either:

- interchange i_1 on shelf j_1k_1 with i_3 on shelf j_2k_2
- interchange i_2 on shelf j_1k_1 with i_4 on shelf j_2k_2
- several checks made for the above interchanges



RemoveLeastProfitable

Choose a random shelf part jk and with probability of 10%, find product i with $x_{ijk} > L_i$ contributing least profit φ_{ijk} and make $x_{ijk} = x_{ijk} - 1$



Experiments and Results

Small Instance: $n = 135, m = 5, p = 3, \text{average}(a_i) = 0.11 mts$

$$\sum_{j=1}^{m}\sum_{k=1}^{p} T_{jk} = 33 mts, L_i = 1 \ \forall i, Ui \in \{2,3,\ldots,11\} \ \forall i$$

Large Instance: $n = 907, m = 38, p = 3, \text{average}(a_i) = 0.13 mts$

$$\sum_{j=1}^{m}\sum_{k=1}^{p} T_{jk} = 248 mts, L_i = 1 \ \forall i, Ui \in \{2,3,\ldots,13\} \ \forall i$$

2. Allocation Phase Results

	Current	**Profit**	**Size**	**Random**
Small	78.64	77.11 (0.07)	71.36 (0.06)	71.80 (0.07)
Large	2463.05	2959.33 (3.55)	2574.38 (3.33)	2513.54 (4.07)

3. Adjustment Phase Results

For each initial arrangement Λ (current, profit, size, random)
Execute **Given Move** during 30 seconds x 30 times

Small Instance

Current	Profit	Size	Random
AdjustInter	MultiShift	MultiShift	MultiShift
MultiShift	AdjustIn	AdjustInter	AdjustInter
AdjustIn	AdjustInter	AdjustIn	AdjustIn
Swap	Swap	Swap	Swap

Large Instance

Current	Profit	Size	Random
AdjustInter	AdjustIn	AdjustIn	MultiShift
MultiShift	MultiShift	MultiShift	AdjustIn
AdjustIn	AdjustInter	AdjustInter	AdjustInter
Swap	Swap	Swap	Swap

Probabilities assigned for the local search:

MultiShift :	35%
AdjustInter :	35%
AdjustIn :	15%
Swap :	15%

For each initial arrangement Λ (current, profit, size, random)
Execute **Adjustment Phase** for *t* seconds x 15 times

Small Instance

Profit Increase		Overall Profit	
Mean	Std.Dev.	Initial	Final
29.06 (37%)	0.77	78.64	107.71
30.93 (40%)	0.93	77.19	108.04
37.19 (52%)	0.83	71.36	108.55
36.25 (50%)	0.52	71.80	108.05

Large Instance

Profit Increase		Overall Profit	
Mean	Std.Dev.	Initial	Final
2386.3 (97%)	51.1	2463.0	4849.3
1869.7 (63%)	27.5	2959.3	4829.1
2391.4 (93%)	23.9	2574.3	4965.8
2191.7 (8%)	36.0	2513.5	4705.2

THE QMC NURSE SCHEDULING PROBLEM

We refer to nurse scheduling or nurse rostering as the process of timetabling staff (allocating nurses to working shifts) over a short period of time (typically few weeks).

Typically, the aim is to produce a roster for the ward over the planning period, so that all hard constraints are satisfied and the violation of soft constraints is minimised.

In particular, we aim to maximise satisfaction of individual preferences.

Hard constraints:

- OneShiftADay
- MaxHours
- MasDaysOn
- MinDaysOn
- Succession
- HardRequest

Soft constraints:

- SoftRequest
- SingleNight
- WeekendBalance
- WeekendSplit
- Coverage
- Coverage Balance

NursePreference(i) = $\{p(i,j) : 1 \leq j \geq \text{NoOfDays}\}$

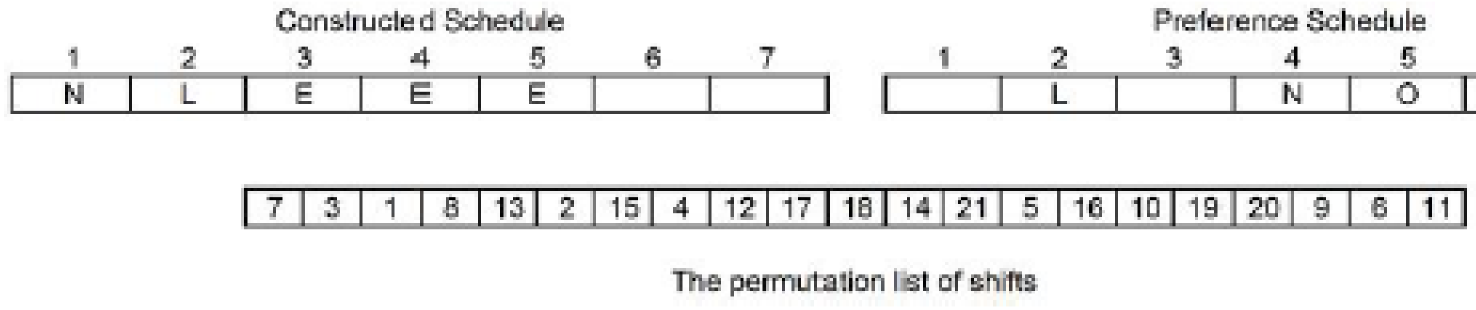
$p(i,j) \in \{\text{AnnualLeave, Any, DayOff, Early, Late, Night}\}$

NurseSchedule(i) = $\{s(i,j) : 1 \leq j \geq \text{NoOfDays}\}$

$s(i,j) \in \{\text{AnnualLeave, DayOff, Early, Late, Night}\}$

Encoding(i) = $\text{Permutation}\{\text{shift} : 1 \leq \text{shift} \geq 3 \cdot \text{NoOfDays}\}$

Ward Schedule = collection of n individual nurse schedules


Constructed Schedule

1	2	3	4	5	6	7
N	L	E	E	E		

Preference Schedule

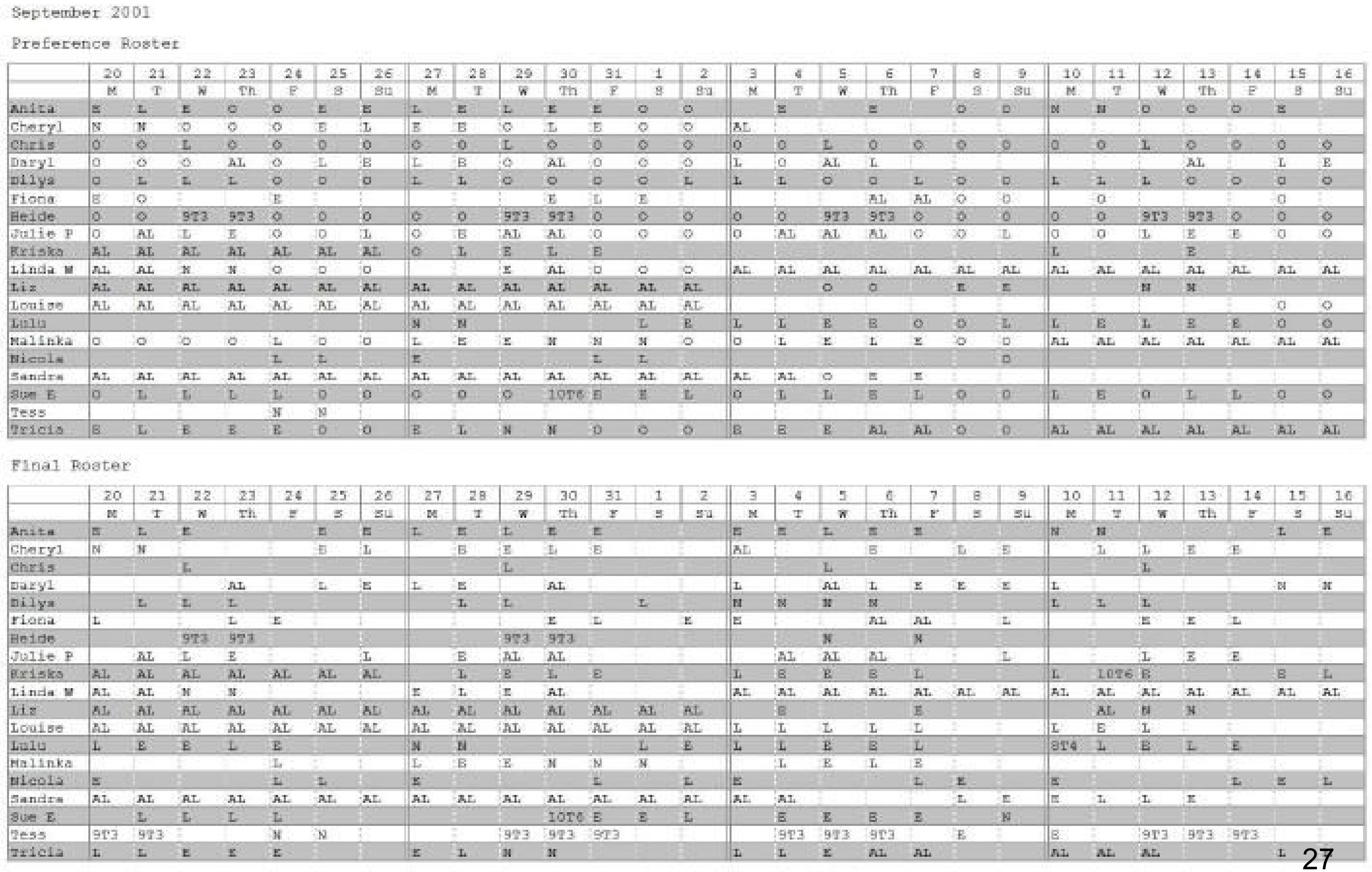
1	2	3	4	5	6	7
	L		N	O		

7	3	1	8	13	2	15	4	12	17	18	14	21	5	16	10	19	20	9	6	11

The permutation list of shifts

Fig. 1. Preference Schedule, Constructed Schedule and Permutation List

Example of preference and constructed schedules



September 2001

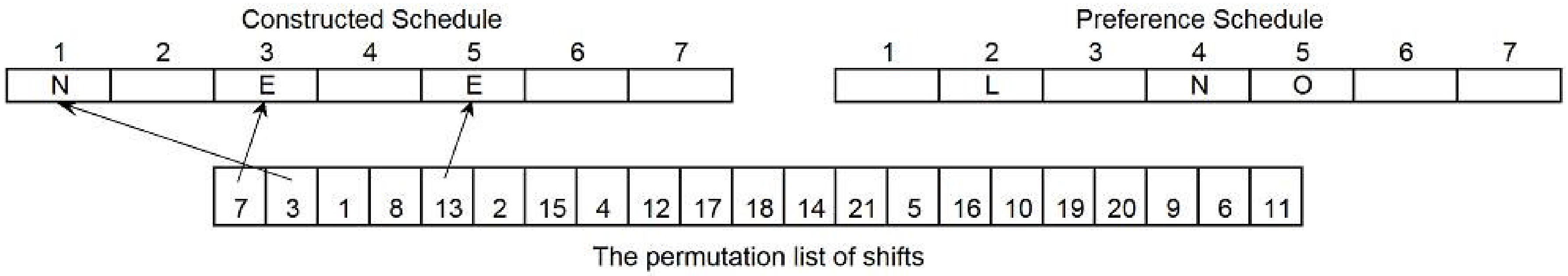
Preference Roster

	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	M	T	W	Th	F	S	Su	M	T	W	Th	F	S	Su	M	T	W	Th	F	S	Su	M	T	W	Th	F	S	Su
Anita	E	L	E	O	O	E	E	L	E	L	E	E	O	O		E		E		O	O	N	N	O	O	O	E	
Cheryl	N	N	O	O	O	E	L	E	E	O	L	E	O	O	AL													
Chris	O	O	L	O	O	O	O	O	O	L	O	O	O	O	O	O	L	O	O	O	O	O	O	L	O	O	O	O
Daryl	O	O	O	AL	O	L	E	L	E	O	AL	O	O	O	L	O	AL	L							AL		L	E
Dilys	O	L	L	L	O	O	O	L	L	O	O	O	O	L	L	L	O	O	L	O	O	L	L	L	O	O	O	O
Fiona	E	O			E						E	L	E					AL	AL	O	O		O				O	
Heide	O	O	9T3	9T3	O	O	O	O	O	9T3	9T3	O	O	O	O	O	9T3	9T3	O	O	O	O	O	9T3	9T3	O	O	O
Julie P	O	AL	L	E	O	O	L	O	E	AL	AL	O	O	O	O	AL	AL	AL	O	O	L	O	O	L	E	E	O	O
Kriska	AL	AL	AL	AL	AL	AL	AL	O	L	E	L	E										L			E			
Linda W	AL	AL	N	N	O	O	O			E	AL	O	O	O	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL
Liz	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL			O	O		E	E			N	N			
Louise	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL													O	O
Lulu								N	N				L	E	L	L	E	E	O	O	L	L	E	L	E	E	O	O
Malinka	O	O	O	O	L	O	O	L	E	E	N	N	N	O	O	L	E	L	E	O	O	AL	AL	AL	AL	AL	AL	AL
Nicola					L	L		E				L	L								O							
Sandra	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	O	E	E									
Sue E	O	L	L	L	L	O	O	O	O	O	10T6	E	E	L	O	L	L	E	L	O	O	L	E	O	L	L	O	O
Tess					N	N																						
Tricia	E	L	E	E	E	O	O	E	L	N	N	O	O	O	E	E	E	AL	AL	O	O	AL	AL	AL	AL	AL	AL	AL

Final Roster

	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	M	T	W	Th	F	S	Su	M	T	W	Th	F	S	Su	M	T	W	Th	F	S	Su	M	T	W	Th	F	S	Su
Anita	E	L	E			E	E	L	E	L	E	E			E	E	L	E	E			N	N				L	E
Cheryl	N	N				E	L		E	E	L	E			AL			E		L	E		L	L	E	E		
Chris			L							L							L							L				
Daryl				AL		L	E	L	E		AL				L		AL	L	E	E	E	L					N	N
Dilys		L	L	L					L	L			L		N	N	N	N				L	L	L				
Fiona	L			L	E						E	L		E	E			AL	AL		L			E	E	L		
Heide			9T3	9T3						9T3	9T3						N		N									
Julie P		AL	L	E			L		E	AL	AL					AL	AL	AL			L			L	E	E		
Kriska	AL	AL	AL	AL	AL	AL	AL		L	E	L	E			L	E	E	E	L			L	10T6	E			E	L
Linda W	AL	AL	N	N				E	L	E	AL				AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL
Liz	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL		E			E				AL	N	N			
Louise	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	L	L	L	L	L			L	E	L				
Lulu	L	E	E	L	E			N	N				L	E	L	L	E	E	L			8T4	L	E	L	E		
Malinka					L			L	E	E	N	N	N			L	E	L	E									
Nicola	E				L	L		E				L		L	E				L	E		E				L	E	L
Sandra	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL	AL				L	E	E	L	L	E			
Sue E		L	L	L	L						10T6	E	E	L		E	E	E	E		N							
Tess	9T3	9T3			N	N				9T3	9T3	9T3				9T3	9T3	9T3		E		E		9T3	9T3	9T3		
Tricia	L	L	E	E	E			E	L	N	N				L	L	E	AL	AL			AL	AL	AL			L	

Decoder



Day=(x−1) div 3 + 1, Shift=(x−1) mod 3

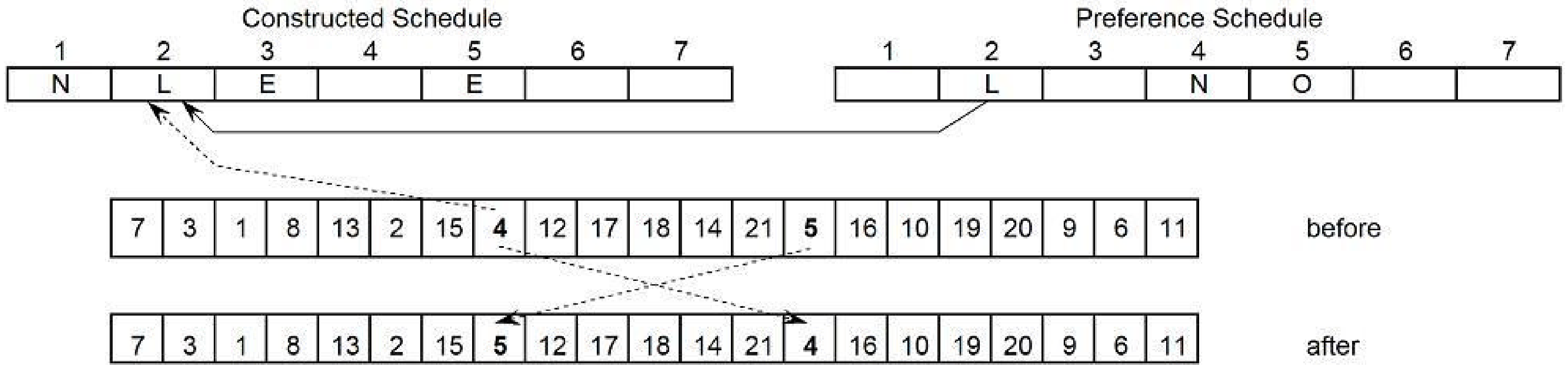
Early, Late, Night correspond to 0, 1, 2 respectively

For example, for x = 7:

D=(7 – 1) div 3 + 1 = 3 S=(7-1) mod 3 = 0

Then assign Early shift to day 3

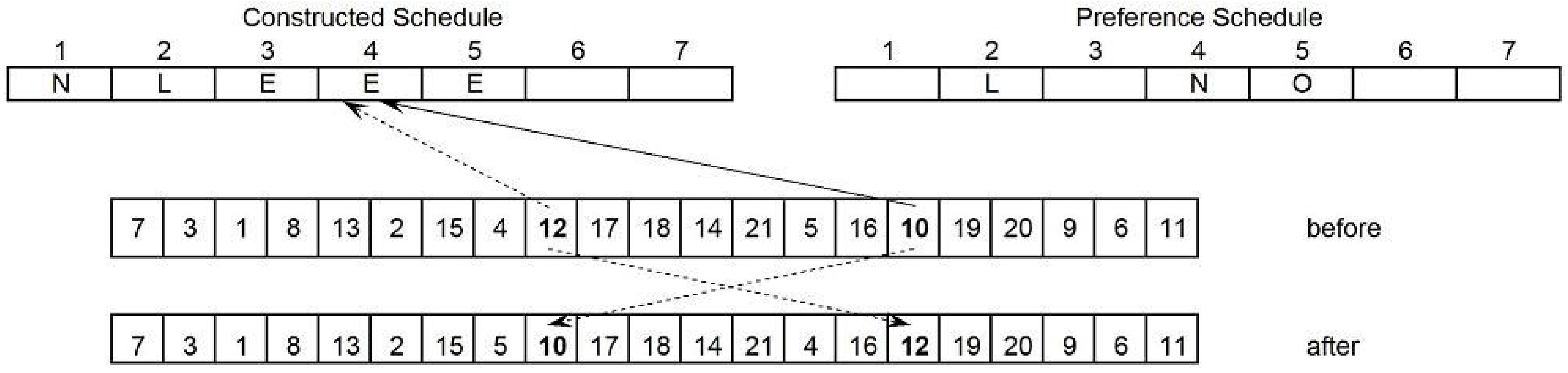
Self-mutation to deal with Succession hard constraint



The permutation list of shifts before and after the decoding process

1. Assigning shift 4 (Early shift to day 2) violates the Succession constraint
2. High Soft-Request-Probability forces decoder to use preference schedule
3. Assigning Late shift to day 2 is allowed
4. Self-mutation to the current permutation list to swap shift 4 and shift 5

Self-mutation to deal with Succession hard constraint



The permutation list of shifts before and after the decoding process

1. Assigning shift 12 (Night shift to day 4) violates the Succession constraint
2. Low Soft-Request-Probability forces decoder to use permutation list
3. Search permutation list and assign shift 10 (Early shift to day 4) is allowed
4. Self-mutation to the current permutation list to swap shift 10 and shift 12

The QMC nurse scheduling is certainly multi-criteria but should it also be treated as a multi-objective problem?

E.g. in product design:

multiple criteria: cost, reliable, profitable, durable

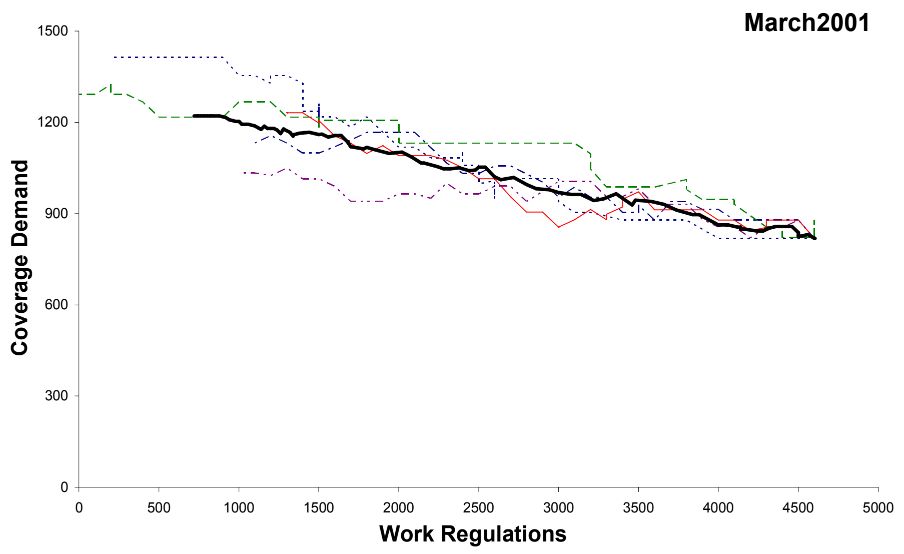
is the above a 4-objective problem in the Pareto sense?

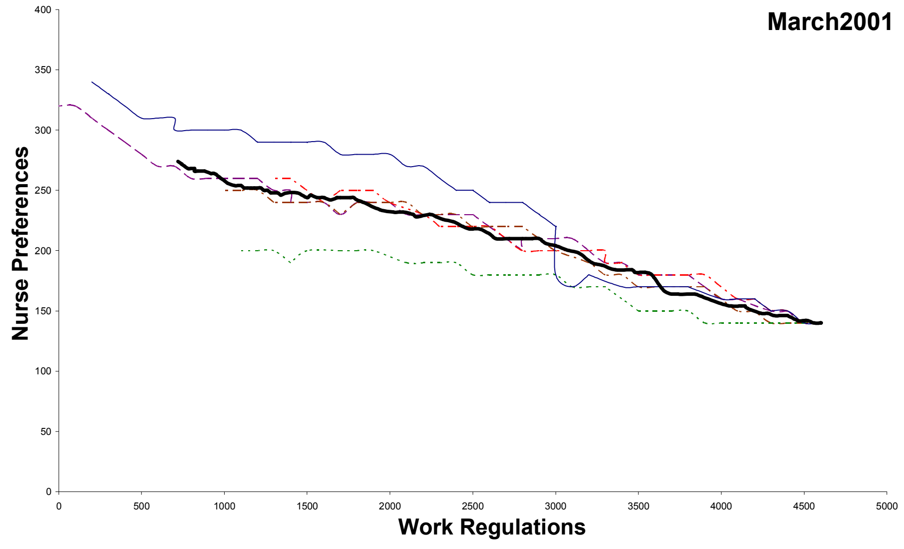
Hard constraints

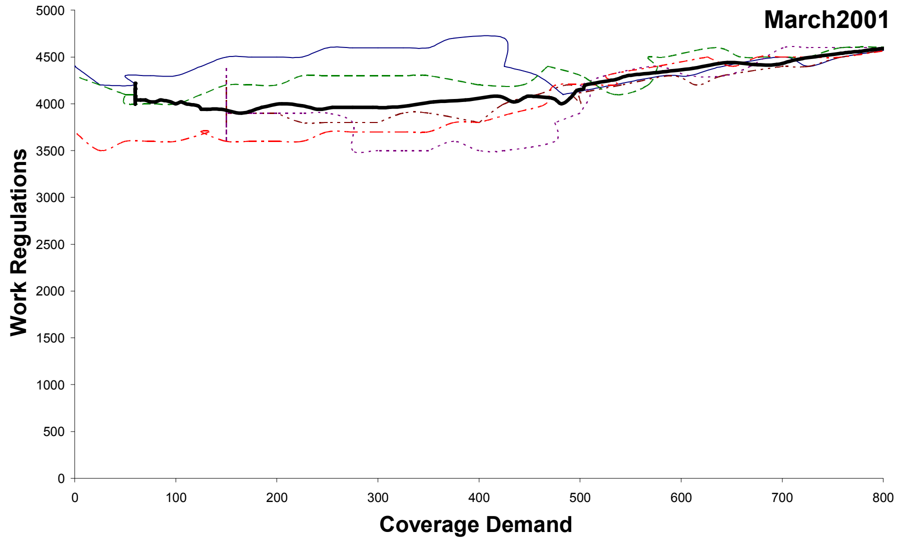
- OneShiftADay
- MaxHours
- MaxDaysOn
- MinDaysOn
- Succession
- HardRequest

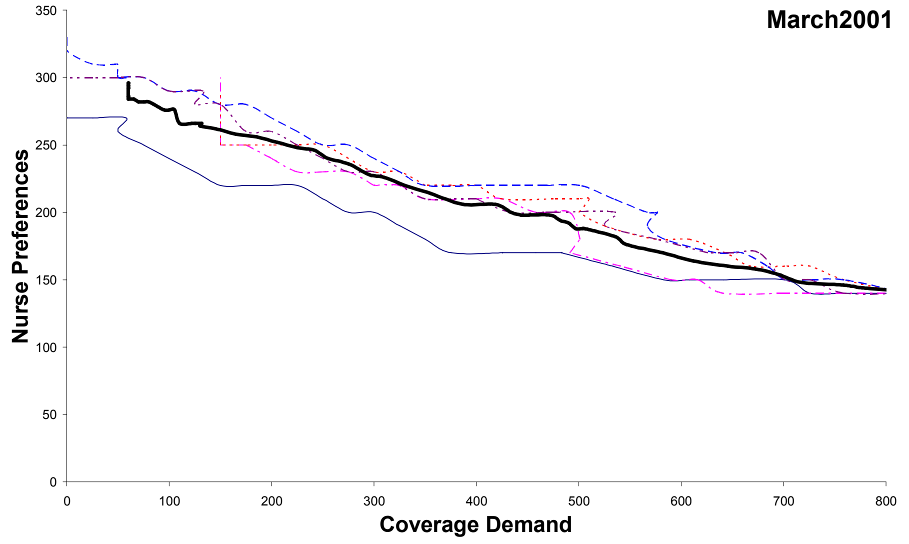
Soft constraints

- SoftRequest	nurse preferences
- SingleNight - WeekendBalance - WeekendSplit	work regulations
- Coverage - Coverage Balance	coverage demand


March2001
Coverage Demand
Work Regulations
0
300
600
900
1200
1500
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000


March2001
Nurse Preferences
Work Regulations
0
50
100
150
200
250
300
350
400
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000


March2001
Work Regulations
Coverage Demand
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
0
100
200
300
400
500
600
700
800


March2001
Nurse Preferences
Coverage Demand
0
50
100
150
200
250
300
350
0
100
200
300
400
500
600
700
800

Overall conflicting nature of criteria in the QMC problem

	Group WR	Group CD	Group NP
Group WR	------	conflict	conflict
Group CD	independent	------	
Group NP	independent	independent	------

Conflict, independence and harmony between criteria may vary during the search

Hence, the search strategy can be adapted to this variation

Some results…

	Initial	SWO	SWO + CS
Mar-01	819	0	0
Apr-01	953	266	205
May-01	1875	618	577
Jun-01	1801	1468	1409
Jul-01	2005	1734	1473
Aug-01	786	330	40
Sep-01	1129	619	495

	ND	Group WR	Group CD	Group NP	AverSimil
Mar-01	6.767	3.390	4.664	2.41	49.2%
Apr-01	7.700	3.027	8.219	2.25	51.6%
May-01	6.933	2.943	15.132	1.93	57.7%
Jun-01	3.933	0.892	38.615	2.50	62.0%
Jul-01	7.900	2.173	38.725	2.68	54.2%
Aug-01	7.200	3.555	7.528	1.90	53.7%
Sep-01	7.467	2.611	18.520	2.07	56.7%

Additional Reading

Chapter 9 of (Michalewicz,2004) and Section 1.5 of (Talbi,2009).

D. Landa-Silva, K.N. Le (2008). *A simple evolutionary algorithm with self-adaptation for multi-objective optimisation.* In: Cotta, C., S. M. & Srensen, K. (eds.) Adaptive and multilevel metaheuristics, Series 'Studies in computational intelligence', Vol. 136, Springer, 133-155.

E. K. Burke, P. de Causmaecker, S. Petrovic, G. Vanden berghe (2001). *Fitness evaluation for nurse scheduling problems.* Proceedings of the 2001 Congress on Evolutionary computation (CEC 2001), 1139-1146.

A. Viana, J.P. de Sousa, M.A. Matos (2003). *GRASP with constraint oriented neighbourhoods: an application to the unit commitment problem*. In: Proceedings of the 5th Metaheuristics International Conference (MIC 2003), 78:1-78:8.

Seminar Activity 6

The purpose of this seminar activity is to discuss the suitability of constraint handling techniques for the GAP variant of Example 2.1.

Do the following:

1. Look at the instances of the GAP variant and estimate the proportion of feasible and infeasible solutions.

2. Discuss the suitability of various constraint handling techniques in the context of this problem, including:

   a) Rejecting infeasible solutions

   b) Enforcing feasibility using specialised moves

   c) Repairing infeasibility using specialised strategies

   d) Penalising infeasible solutions

   e) Using constraint-oriented neighbourhoods