The Template Design Problem: A Perspective with Metaheuristics

David Rodríguez[1] Carlos Cotta[2] Antonio J. Fernández[2]

[1]Laboratorio de Computación de Alto Rendimiento, Universidad Nacional Experimental del Táchira (UNET), Venezuela

[2]Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga, Spain

The Template Design Problem

The TDP

The Template Design Problem (TDP) is a constrained combinatorial problem arising in different production settings in which product demands must be fulfilled within given tolerance, minimizing the production cost.

A two-level optimization setting

Solving the TDP involves a two-level optimization approach in which metaheuristics tackle a design problem, which is then submitted to an integer programming solver to obtain the actual production schedule.

Variations to be produced

Formulation of the TDP

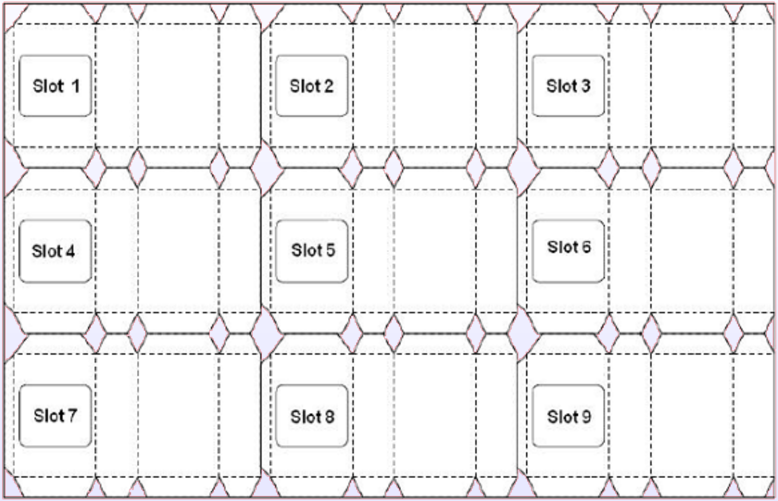
The TDP

Let there be v design variations we want to produce. For each of them ($1 \leqslant i \leqslant v$) we know:

- the production demand Q_i,
- the tolerance to overproduction (u_i) and to underproduction (l_i).

We have to design a set of T templates, each of them with s slots, so that we minimize the production cost.

A template example



An ILP Model

Objective

$$\min \sum_{i=1}^{v} (U_i + O_i)$$

Subject to

$$\sum_{j=1}^{T} p_{ij} R_j + U_i - O_i = Q_i, \quad 1 \leqslant i \leqslant v$$
$$\sum_{j=1}^{T} p_{ij} R_j \geqslant (1 - l_i) Q_i, \quad 1 \leqslant i \leqslant v$$
$$\sum_{j=1}^{T} p_{ij} R_j \leqslant (1 + u_i) Q_i, \quad 1 \leqslant i \leqslant v$$
$$R_j \geqslant 0 \quad 1 \leqslant j \leqslant T$$
$$U_i, O_i \geqslant 0 \quad 1 \leqslant i \leqslant v$$

An example
[Proll, Smith, 1999]

Let there be the following 7 variations:

variation	demand $(Q_i/1000)$
Liver	250
Rabbit	255
Tuna	260
Chicken Twin	500
Pilchard Twin	500
Chicken	800
Pilchard	1,100

Let us assume templates have $s = 9$ slots, and a tolerance of 10%.

An example
[Proll, Smith, 1999]

At least two templates are required to obtain a feasible solution:

	templates	
variation	T_1	T_2
Liver	0	1
Rabbit	0	1
Tuna	0	1
Chicken Twin	0	2
Pilchard Twin	0	2
Chicken	2	2
Pilchard	7	0
pressings	157,143	250,000

Candidate solutions

Solutions are represented as a $v \times T$ matrix $P = \{p_{ij}\}$. All entries in such a matrix are non-negative and subject to

$$\sum_{i=1}^{v} p_{ij} = s, \quad 1 \leqslant j \leqslant T$$

Additionally, there exist production constraints modelled as soft constraints in the evaluation function.

Neighborhood structure

A minimal change can be done in a solution by means of reassigning a single slot to a different variation, subject to hard feasibility constraints, i.e., let $P = \{p_{ij}\}$, then

$$\mathcal{N}(P) = \{P' \mid \exists j, i, i' : p_{ij} > 0, p_{i'j} < s, p'_{ij} = p_{ij} - 1, p'_{i'j} = p_{i'j} + 1\}$$

Notice $|\mathcal{N}(P)| \in O(Tv^2)$.

Objective function

A 3-level stratified fitness function is used:

1. solve the ILP model. If a solution is found, use the optimal ILP optimal value as objective value to be minimized.
2. if no solution is found, relax the overproduction constraint, solve again and add a penalty term.
3. if no solution is found either, relax the underproduction constraint, aolve again and add another penalty term

Initialization

Initialization can be done at random, or following some ad-hoc heuristic.

The heuristic can be viewed as a biased random initialization in which variations are given slots with a probability proportional to the fraction of the total demand they represent.

A correction mechanism is also used to avoid that some variation is associated to no slot. This means to enforce the constraint

$$\forall i \in V : \sum_{j=1}^{T} p_{ij} > 0. \quad (1)$$

Initialization

Initialization pseudocode

```
begin
    p ← InitializeTemplate(T, V);
    // Computes the fraction of the demand for each variation.
    pq ← ComputeProbabilities(Q);
    for i ← 1 to T do
        // Fills the i-th template according to pq.
        RouletteFill(p, pq, i, V);
        // Computes a heuristic number of pressings for the i-th
        // template and updates probabilities according to the
        // remaining demand.
        R ← min_{j=1...V} {(1 − L_j)Q_j/p_ij | p_ij > 0};
        for j ← 1 to V do
            Q_j ← Q_j − R * p_ij;
        end for
        pq ← ComputeProbabilities(Q);
    end for
end
```

Experiments

Algorithms considered:

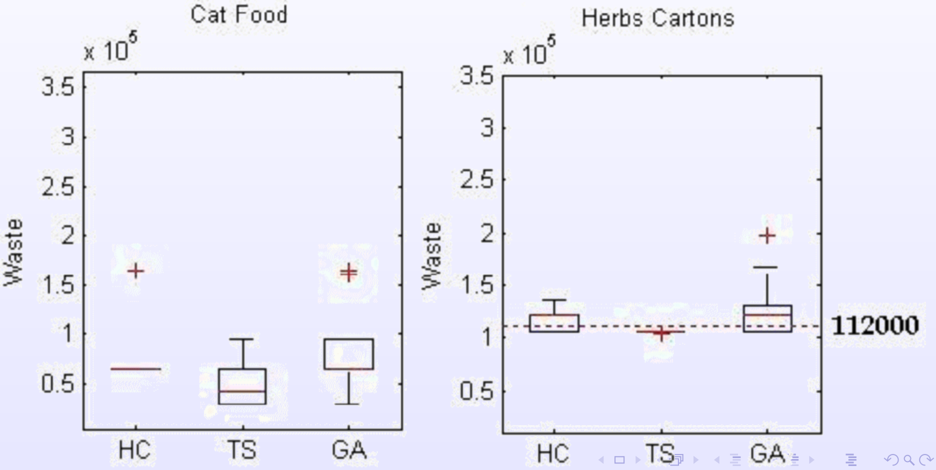
1. Local-search: steepest ascent hill climbing and tabu search. TS features randomized tabu tenure, intensification after n_ι evaluations without improvement, and improvement of the best solution so-far as aspiration criterion.
2. Population-based: an evolutionary algorithm featuring slot-reassignment for mutation, and a uniform template-based crossover.

Tree instances considered:

1. Cat food: 7 variations and 9 slots per template.
2. Herbs cartons: 30 variations and 42 slots per template.
3. Magazine Inserts: 50 variations and 40 slots per template.

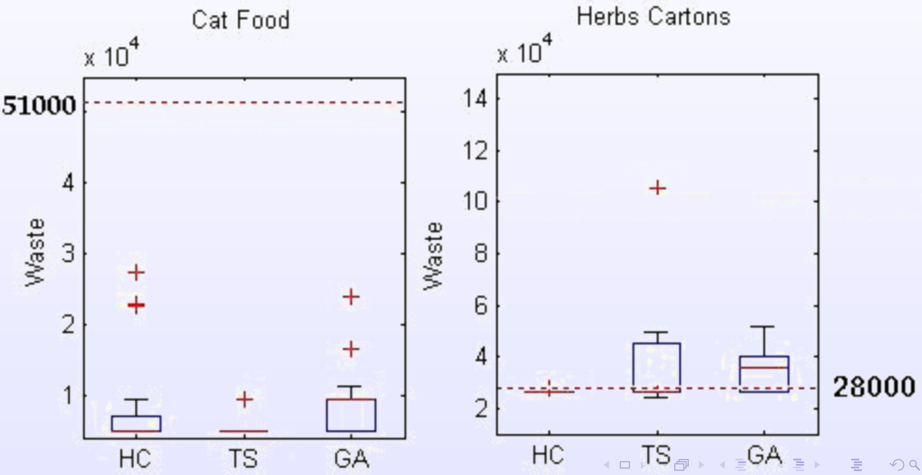
Results

2 Templates



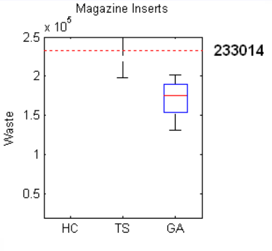
Results

3 Templates



Results
4 Templates

Feasible solutions for the stage Magazine Inserts, for 4 templates.



Note that for HC and TS algorithms did not find feasible solutions for this scenario

Conclusions

Metaheuristic approaches provide encouraging results for the TDP.

Ongoing work on memetic algorithms for the problem. LS too costly for a fully Lamarckian approach.

Scalability is also an issue. Further work required in smart recombination operators.