Towards an Evolutionary Tool for the Allocation of Supermarket Shelf Space

Anna I Esparcia-Alcázar[*]
Lidia Lluch-Revert
Ken C Sharman
Instituto Tecnológico de Informática
Edif. 8G Acc. B,
Ciudad Politécnica de la Innovación
46022 Valencia, Spain
anna, lillure, ken@iti.upv.es

José Miguel Albarracín-Guillem
Marta E Palmer-Gato
Departamento de Organización de Empresas,
Econ. Financiera y Contabilidad
Universidad Politécnica de Valencia
Camino de Vera s/n
46022 Valencia, Spain
jmalbarr, marpalga@omp.upv.es

ABSTRACT

In this paper we set the first steps towards the development of a commercially viable tool that uses evolutionary computation to address the Product to Shelf Allocation Problem (P2SAP). The problem is described as that of finding the numbers and locations of modules to allocate to particular products in a shop, fulfilling at the same time a number of constraints. We first justify the use of evolutionary algorithms in this problem in the bad scalability properties shown by exact methods. Then we proceed, from simpler to more complex versions of the problem, to describe different encodings, fitness functions and evolutionary operators that are suited to the problem. The variations described are tested on five different problem configurations: three with one shelf, one with two shelves and one with eight shelves. In all cases acceptable results can be obtained in a very short timescale, although there is much work to be done on the subject.

Categories and Subject Descriptors: F.2.2 [Theory of Computation]: Nonnumerical Algorithms and Problems

General Terms: Algorithms.

Keywords: Shelf-space allocation, evolutionary algorithms.

1. INTRODUCTION

The problem of allocating space to a particular product in a shop is typically addressed in the literature as that of deciding what combination of products will yield the maximum profit. In general, the most commonly employed methods try and measure the impact on the customer of the relationship between allocated space and sales [1][2][5][7]. The aim is to find the allocation of space that maximises the profit.

[*]Corresponding author.

In [9] we can find an extensive literature survey of both exact methods and heuristics applied to the allocation of shelf space to each product, although the authors point out the lack of academic work on this subject. Also, the models found are complex and have practical limitations. Further, the simplifications used make retail application unrealistic. All these methods face the problem of incorporating real world cases, due to the existing complexity and the multitude of variables.

In the real business case that we were presented with, however, the maximisation of the profit and the actual allocation of products to shelves were solved at different levels. At the top level, the management determine what are the optimal (profit-wise) lengths of shelves to allocate to each product, called the *category standards.* The set of category standards that configures a shop is termed the *standard shop.* At the lower level, the shop planners must take this "ideal" shop configuration and adapt it to the actual space available. Their objective is to allocate lengths of shelves to particular products, given a real shop layout and the standard shop requirements. The problem they face is the lack of suitable tools to achieve this.

The aim of this paper is to present an evolutionary tool that can precisely do this. On the one hand, the method can handle realistic problems (both in terms of size and complexity); on the other, it is at the same time operationally viable, in that it can provide results within reasonable timescales. At the same time, the tool can provide solutions that are easily understandable to the shop planner.

The paper is laid out as follows. In Section 2 the problem is described, together with the exact and evolutionary approaches to address it. In Section 3 the experiments are detailed and Section 4 summarises the conclusions and further work.

2. THE PRODUCT TO SHELF ALLOCATION PROBLEM (P2SAP)

2.1 Problem description

In this paper we will concentrate on finding actual values of lengths per product that are close enough to the ideal values, and locating them so as to fulfill other requirements. This we refer to as the Product to Shelf Allocation Problem (P2SAP), to distinguish it from the Shelf-Space Allocation Problem (SSAP) described in [9]. The P2SAP that we describe here arose as part of the expansion process of a Spanish supermarket chain, whose managers were interested in opening a significant number of new supermarkets in a short period of time. When a new store is opened, the shop planners are given the standard shop requirements. In practice, this is a list of all categories (or products) plus their corresponding category standards. A shelf is divided into modules of given length, so the category standard can be expressed as a length in metres or as an integer number of modules, with a minimum of one. Some categories can also have a specification of number of modules by which their category standard can be increased or decreased if necessary. Because the size and layout of different premises can vary, the standard shop requirements cannot be fulfilled exactly, so for some categories the number of allocated modules have to be increased or decreased.

There are also other constraints for the placing of categories. Firstly, categories are classified into groups (e.g., baby milk belongs to the generic group of baby products) so categories in the same group must be placed together. Groups should be kept as cohesive as possible; it does not make sense to put a module containing tinned beans at the end of a shelf full of toiletries, only because that is what the standard number of modules for that category specifies.

Next, categories and groups can have *affinities* or *disparities* between them. For instance, baby products are disparate, or adverse, to pet food because, seemingly, customers do not like seeing the two things together. Groups containing food products are affine to each other, but indifferent to housewares.

Finally, some groups must be placed near *reference points*. For instance, bakery products must be placed near the oven (even though they are not manufactured in store) or expensive spirits must be placed near the checkouts, so that the cashiers can keep an eye on them.

With all these constraints and the standard shop requirements, the shop planner can take several days to come up (by hand) with a layout for the new shop. If many shops have to be opened at the same time, this is not acceptable. Also, for existing shops, there is the problem of new categories being introduced everyday, which implies frequent reorganisation.

2.2 Definitions

Given a set of K categories (or products) that can be grouped into a set of G groups, and a set of S shelves, each one divided into M_i, $i = 1, \ldots, S$ modules, with the total number of modules $M = \sum_{i=1}^{s} M_i \geq K$.

The P2SAP consists of finding a matrix $\mathbf{X}$ representing an allocation of categories to modules,

$$x_{i,k} = \begin{cases} 1 & \text{if category } c_i \text{ is assigned to module } m_k \\ 0 & \text{otherwise} \end{cases}$$

subject to the following constraints:

- *Space* available. This is a hard constraint that can be represented as
  $\sum_{i=1}^{K} \sum_{u=1}^{M} x_{i,u} = M$
- *Standard shop* requirements. We will capture these in a matrix as follows

$$\mathbf{SS} = \begin{pmatrix} std_1 & min_1 & max_1 & p_1 \\ std_2 & min_2 & max_2 & p_2 \\ \vdots & \vdots & \vdots & \vdots \\ std_K & min_K & max_K & p_K \end{pmatrix} \quad (1)$$

  where std_i, min_i and max_i are the standard, minimum and maximum number of modules to allocate to unit i, respectively, (where a unit can be a category or a group) and p_i is the preference for the increase in modules

- *Affinity* constraints. We will represent these by a $K \times K$ *affinity matrix*, $\mathbf{A}$, where each component $a_{i,j}$ denotes the affinity between categories i and j, as follows:

$$a_{i,j} \begin{cases} < 0 & \text{if } c_i \text{ and } c_j \text{ are adverse} \\ = 0 & \text{if } c_i \text{ and } c_j \text{ are indifferent} \\ > 0 & \text{if } c_i \text{ and } c_j \text{ are affine} \end{cases}$$

- Location with respect to reference points
- *Group cohesion.* This and the previous one are softer constraints and we will see how to handle them below.

The first two constraints relate to the number of modules to allocate to each category, while the last three relate to their relative position. So we have a dual objective to achieve: on the one hand, to find the "ideal" number of modules (as opposed to the "standard" number); on the other hand, to find the ideal allocation of categories to the modules. However, in this first approach to the problem we will use a simple heuristic to attain the first objective and will concentrate on the second objective.

For this we will need a further definition: an $M \times M$ *distance matrix*, $\mathbf{D}$, where each component $d_{i,j}$ denotes the distance between units i and j, where a unit can be either a category, a module or a group, depending on the problem configuration.

2.3 Exact approaches to the P2SAP

The first approach we can take to this problem is the complete enumeration of the solutions. Since essentially what we are looking for is a permutation of the K categories, we can find an algorithm that generates all the possible $K!$ permutations, evaluates them and chooses the best. However, this is only feasible for really small versions of the problem. For $K = 16$ and assuming a time of $1\mu s$ per operation, just generating the solutions would take 8 months; for $K = 17$ the time would be above 10 years.

A second approach to the P2SAP is to model it as a Mixed Integer Quadratic Assignment Problem (MIQAP) using an objective function as follows:

$$f = \frac{1}{2} \sum_{i=1}^{K} \sum_{j=1}^{K} \sum_{k=1}^{M} \sum_{l=1}^{M} a_{ij} \cdot d_{kl} \cdot x_{ik} \cdot x_{jl}, \quad j > i, \quad k \neq l \quad (2)$$

The aim is to find combinations of the x_{ik} and x_{jl} that minimise f, subject to the constraints shown above. In our experiments, for a simple problem ($S = 1$, $M = 10$, $K = 5$) a solution was obtained in a few minutes. However, for a more complex problem ($S = 1$, $M = 17$, $K = 10$) the run time was of the order of 30 to 40 hours. For even more complex problems we could not obtain a solution with the software employed[1]. In view of these figures and given that, according to [10], an average supermarket of medium size has between 5000 and 7000 categories, it seems obvious that a different approach to their allocation must be taken. The choice was an evolutionary approach.

2.4 The evolutionary approach to the P2SAP

For simplicity reasons we decided not work with the matrix **X** and tried different codings instead. The coding varied with each problem configuration and will be explained below. The same goes for the fitness function employed.

The evolutionary process started by generating a random population of individuals or chromosomes representing solutions to the problem at hand. Then evolution proceeded employing the following operators:

- `reOrderXover` : This is a variant of the standard crossover operator used when the chromosomes are permutations of a series of natural numbers and therefore each number can appear only once. Two parents are selected. The genes (components) of each parent are reordered from the (random) crossover point in the same order as they appear in the other parent.
- `swap` : The values of two genes selected at random in a chromosome are swapped
- `shift(n)` : The genes in a chromosome are shifted n (a random value) positions to the left if $n < 0$ or to the right if $n > 0$
- `shuffle(n)`: Given a random number n, select randomly a segment of n genes within the chromosome and shuffle their values.

3. EXPERIMENTS

3.1 One shelf - P2SAP1

This is the simplest version of the problem. We considered three variants:

- P2SAP1.1 - The number of modules equals the number of categories ($M = K$)
- P2SAP1.2 - The number of modules is greater than the number of categories ($M > K$)
- P2SAP1.3 - As P2SAP1.2, but with a *reference point*

As said above, we did not work with the matrix **X**, but rather collapsed it into a single vector **chrom**, the *chromosome* or individual. In general the chromosomes are vectors of natural numbers which in the simpler versions of the problem represent the categories.

$\boxed{\text{P2SAP1.1}}$

For this simple case we did not consider groups, or rather, we assumed that each category belongs to its own group. To represent affinities between categories, the components of **A** will take values of 1, -1 or 0.

We implemented an evolutionary algorithm in which the chromosomes are coded as vectors of length M where each component i stores the category allocated to module i. Since $M = K$ we always have a one-to-one correspondence between modules and categories.

Parameters of the evolutionary algorithm	
Population size	100 (P2SAP1), 200 (P2SAP2), 400 (P2SAPn)
Selection method	Tournament
Tournament size	5 (P2SAP1 & 2), 10 (P2SAPn)
Mutation probability	$1/chromLength$
Diversity preservation mechanism	Restart + seeding with best when 80% individuals are equal

Table 1: Parameters of the evolutionary algorithm employed in the experiments. The values of the population and tournament sizes were chosen after experimentation. Following [6], the mutation probability equals the inverse of the chromosome length.

We considered modules of unit length and defined the distance between two consecutive modules as equal to 1. The fitness function was defined as follows:

$$f = \frac{1}{2}\sum_{i=1}^{K}\sum_{j=1}^{K}(d_{i,j}\cdot|a_{i,j}|)^{sgn(a_{i,j})} \quad (3)$$

where

$d_{i,j}$	is the distance between modules m_i and m_j
$a_{i,j}$	is the affinity between the categories allocated to modules m_i and m_j
$sgn(a_{i,j})$	$= \begin{cases} sign(a_{i,j}) & \forall a_{i,j} \neq 0 \\ 0 & a_{i,j} = 0 \end{cases}$

and the objective was to *minimise* f.

We took $M = K = 10$ and used a randomly generated affinity matrix, as follows

$$A = \begin{pmatrix} - & 1 & -1 & 0 & 0 & 1 & 1 & 0 & -1 & 1 \\ 1 & - & 0 & 0 & 1 & 1 & 1 & -1 & 0 & 1 \\ -1 & 0 & - & 1 & 0 & 1 & -1 & 0 & 1 & -1 \\ 0 & 0 & 1 & - & -1 & -1 & -1 & 0 & -1 & -1 \\ 0 & 1 & 0 & -1 & - & -1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & -1 & -1 & - & 0 & 1 & 0 & -1 \\ 1 & 1 & -1 & -1 & 1 & 0 & - & 1 & 1 & -1 \\ 0 & -1 & 0 & 0 & 0 & 1 & 1 & - & 1 & 0 \\ -1 & 0 & 1 & -1 & 1 & 0 & 1 & 1 & - & 1 \\ 1 & 1 & -1 & -1 & 0 & -1 & -1 & 0 & 1 & - \end{pmatrix}$$

This means that

c_1	is affine to c_2, c_6, c_7 and c_{10} and adverse to c_3 and c_9
c_2	is affine to c_1, c_5, c_6, c_7 and c_{10} and adverse to c_8
c_3	is affine to c_4, c_6, and c_9 and adverse to c_1, c_7 and c_{10}
c_4	is affine to c_3 and adverse to c_5, c_6, c_7, c_9 and c_{10}

and so on. We ran the evolutionary algorithm 20 times. After 3 seconds, the best result obtained was always:

$$\mathbf{chrom} = \begin{pmatrix} 4 & 3 & 6 & 8 & 9 & 7 & 5 & 2 & 10 & 1 \end{pmatrix}$$

or its symmetric:

$$\mathbf{chrom} = \begin{pmatrix} 1 & 10 & 2 & 5 & 7 & 9 & 8 & 6 & 3 & 4 \end{pmatrix}$$

[1]CPLEX version 9.1 running on a Pentium IV 3.0 GHz with 1 GB RAM.

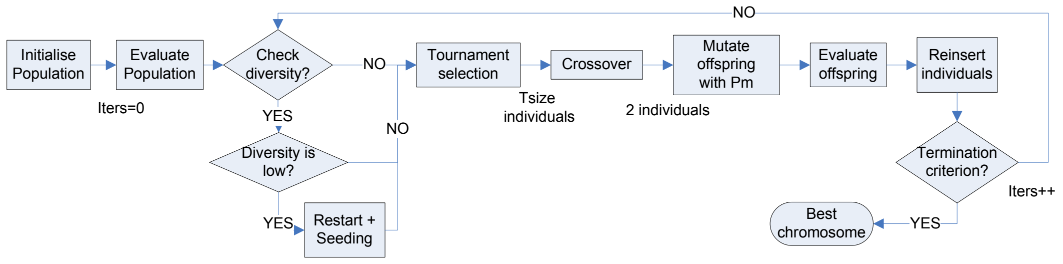

Figure 1: The evolutionary algorithm employed in the experiments

resulting in a shelf layout as follows:

c_4	c_3	c_6	c_8	c_9	c_7	c_5	c_2	c_{10}	c_1

or, conversely,

c_1	c_{10}	c_2	c_5	c_7	c_9	c_8	c_6	c_3	c_4

Analysing the result, we can see that c_4, which is adverse to 4 categories, is placed at the end of the shelf, next to the only other category it is affine to. In other cases, compromises must be reached. This is for instance the case of c_6, which wants to be close to c_3 but far from c_4. We must point out that the affinity matrix was obtained at random and in reality this kind of situation may not arise because there are not so many affinities and disparities between categories, most of them being indifferent to one other.

$\boxed{\text{P2SAP1.2}}$

When $M > K$ we must also allocate the number of modules that correspond to each category and not only their position. This will be given by an *allocation vector*, $\mathbf{h}$. In order to calculate $\mathbf{h}$ in a way that satisfies the requirements of the standard shop, we will employ the category standards (or standard shop) matrix $\mathbf{SS}$ defined in Equation 1. In particular, we will work with the last three columns of this matrix. The last column defines the preference of each category, the highest value corresponding to the category of highest preference (i.e., the one that should be allocated more modules). The second and third columns specify the minimum and maximum number of modules per category respectively.

The allocation procedure is as follows. First of all, we allocate the minimum number of modules to each category. Next we allocate the remaining modules using a variant of the *highest averages method* (or d'Hondt method[2]). The first spare module is allocated to the category with the highest preference. In the next round, the category that was allocated a module gets its preference divided by 2. In general, if a category has been allocated n spare modules, its preference to get the next one (assuming its maximum has not been reached) is $\frac{p_0}{n+1}$, where p_0 is the initial preference. The process ends when all the spare modules have been allocated.
For our experiments we used $M = 10$ and $K = 8$ and a category standard matrix as follows:

[2]This method is named after 19th century Belgian mathematician Victor d'Hondt and is used in the European Union, as well as several European countries, for the allocation of seats in parliament; see, for instance, [4].

$$\mathbf{SS} = \begin{pmatrix} 5 & 1 & 4 \\ 1 & 1 & 4 \\ 2 & 1 & 4 \\ 4 & 1 & 4 \\ 7 & 1 & 4 \\ 8 & 1 & 4 \\ 3 & 1 & 4 \\ 6 & 1 & 4 \end{pmatrix} \tag{4}$$

where the first column has been omitted because it was not used. Hence, categories c_6 and c_5 have the highest preference and c_2 and c_3 the lowest. According to the highest averages method, the allocation of modules to categories would then be a vector $\mathbf{h}$ as follows:

$$\mathbf{h} = \begin{pmatrix} 1 & 1 & 1 & 1 & 2 & 2 & 1 & 1 \end{pmatrix}$$

meaning all categories get allocated one module, except c_5 and c_6, which get two.
The affinity values will be given by

$$a_{i,j} = \begin{cases} -1 & \text{if } c_i \text{ and } c_j \text{ belong to adverse groups} \\ 0 & \text{if } c_i \text{ and } c_j \text{ belong to indifferent groups} \\ 1 & \text{if } c_i \text{ and } c_j \text{ belong to affine groups} \\ 2 & \text{if } c_i \text{ and } c_j \text{ belong to the same group} \\ 3 & \text{if } c_i \text{ and } c_j \text{ belong to the same group} \\ & \text{and are affine to each other} \end{cases}$$

We chose the following combination of groups:

g_1:	categories c_1, c_2, c_3 and c_4
g_2:	categories c_5 and c_6
g_3:	categories c_7 and c_8

where groups g_1 and g_2 are adverse; group g_3 is indifferent to g_2 and affine to g_1. Within group g_1, categories c_1 and c_3 are affine and the rest are indifferent. This results in an affinity matrix as follows:

$$A = \begin{pmatrix} - & 2 & 3 & 2 & -1 & -1 & 1 & 1 \\ 2 & - & 2 & 2 & -1 & -1 & 1 & 1 \\ 3 & 2 & - & 2 & -1 & -1 & 1 & 1 \\ 2 & 2 & 2 & - & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & - & 2 & 0 & 0 \\ -1 & -1 & -1 & -1 & 2 & - & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & - & 2 \\ 1 & 1 & 1 & 1 & 0 & 0 & 2 & - \end{pmatrix} \tag{5}$$

The chromosome is encoded as a vector whose length equals the number of categories, but in this case this is not the same as the number of modules. The chromosome indicates the order of the categories in the shelf and we will use the allocation vector given by the highest averages method to determine the layout of the actual shelf.

We ran the experiment 80 times with a population size of 100 and a tournament size of 5. The run time was 60 seconds, but typically within two seconds a minimum fitness of 52.5845 was reached. This corresponds to 32 similar solutions, some of which are given here:

$$\mathbf{chrom} = \left(\boxed{6 \quad 5} \quad \boxed{8 \quad 7} \quad \boxed{2 \quad 3 \quad 1 \quad 4} \right)$$
$$\mathbf{chrom} = \left(\boxed{5 \quad 6} \quad \boxed{7 \quad 8} \quad \boxed{1 \quad 3 \quad 2 \quad 4} \right)$$
$$\mathbf{chrom} = \left(\boxed{4 \quad 3 \quad 1 \quad 2} \quad \boxed{8 \quad 7} \quad \boxed{6 \quad 5} \right)$$
$$\mathbf{chrom} = \left(\boxed{4 \quad 2 \quad 3 \quad 1} \quad \boxed{8 \quad 7} \quad \boxed{6 \quad 5} \right)$$

where we have boxed those categories belonging to the same group to show that the algorithm has placed them together. Also, categories c_1 and c_3 are next to each other in all the solutions, groups g_1 and g_2 are always apart and groups g_1 and g_3 are always together.

The actual shelf layout corresponding to, say, the first solution, given the allocation of modules to categories obtained by the highest averages algorithm, would be as follows:

c_6	c_6	c_5	c_5	c_8	c_7	c_2	c_3	c_1	c_4

i.e. there are two modules for categories c_5 and c_6 and one for the rest.

It is worth noticing that the problem could be greatly simplified if instead of placing the categories we concentrated on placing the groups. We will make use of this idea later on.

$\boxed{\text{P2SAP1.3}}$

In this problem a reference point is introduced, which we will assume is placed at the header of the shelf, on the left hand side. We simulate this as a fictitious module, m_0, plus a fictitious category, c_0, that will always be placed in that module. We introduce an extra row and column in the affinity and distance matrices. The categories that must be close to the reference point will have an affinity value of 2 to the fictitious category. This is similar to the affinity value between categories belonging to the same group. The affinity matrix in this case was the same as in Equation 5 but modified adding one new row and column for the fictitious category. In these, all the values are zero except those corresponding to c_7 and c_8, which equal 2, meaning categories c_7 and c_8 must be placed close to the reference point.

$$\mathbf{A} = \begin{pmatrix} - & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 \\ 0 & - & 2 & 3 & 2 & -1 & -1 & 1 & 1 \\ 0 & 2 & - & 2 & 2 & -1 & -1 & 1 & 1 \\ 0 & 3 & 2 & - & 2 & -1 & -1 & 1 & 1 \\ 0 & 2 & 2 & 2 & - & -1 & -1 & 1 & 1 \\ 0 & -1 & -1 & -1 & -1 & - & 2 & 0 & 0 \\ 0 & -1 & -1 & -1 & -1 & 2 & - & 0 & 0 \\ 2 & 1 & 1 & 1 & 1 & 0 & 0 & - & 2 \\ 2 & 1 & 1 & 1 & 1 & 0 & 0 & 2 & - \end{pmatrix} \quad (6)$$

The problem can then be reduced to the previous case, with the restriction that c_0 will always be located in m_0.

For this example we chose $M = 20$ and two values of K, $K = 8$ and $K = 10$. For $M = 20$ and $K = 8$ and a preference matrix as given by Equation 4, the highest averages algorithm provides a module allocation h as follows:

$$\mathbf{h} = \left(3 \quad 1 \quad 2 \quad 3 \quad 3 \quad 3 \quad 2 \quad 3 \right)$$

which means c_2 gets one module, c_3 and c_7 get 2 modules each and the rest get 3 modules each. We ran the evolutionary algorithm 30 times with a population size of 100 and a tournament size of 5. The run time was 15 seconds (increasing the time does not provide better results). When the diversity is low (80% of individuals are the same), the population is reinitialised and seeded with the best result so far (see Figure 1). The best results obtained were as follows:

$$\mathbf{chrom} = \left(0 \quad 8 \quad 7 \quad 4 \quad 2 \quad 3 \quad 1 \quad 6 \quad 5 \right)$$
$$\mathbf{chrom} = \left(0 \quad 8 \quad 7 \quad 4 \quad 2 \quad 3 \quad 1 \quad 5 \quad 6 \right)$$

the former resulting in an actual shelf layout as follows:

c_8	c_8	c_8	c_7	c_7	c_4	c_4	c_4	c_2	c_3	c_3	c_1	c_1	c_1	c_6	c_6	c_6	c_5	c_5	c_5

with categories c_7 and c_8 placed next to the header of the shelf, as was required.

For $M = 20$ and $K = 10$ we add two extra categories, c_9 and c_{10}, to group g_3. The category standards matrix was the same as in Equation 4 adding two rows at the end to represent that c_9 and c_{10} have now the highest preference. This gives a module allocation vector $\mathbf{h}$ of :

$$\mathbf{h} = \left(2 \quad 1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 1 \quad 2 \quad 3 \quad 3 \right)$$

The affinity matrix was modified adding two new rows and columns for the new categories, but also the affinity values of c_7 and c_8 to the reference point were increased to 3.

The best results obtained after 50 runs were

$$\mathbf{chrom} = \left(0 \quad 8 \quad 7 \quad 9 \quad 10 \quad 2 \quad 3 \quad 1 \quad 4 \quad 6 \quad 5 \right)$$
$$\mathbf{chrom} = \left(0 \quad 8 \quad 7 \quad 10 \quad 9 \quad 2 \quad 3 \quad 1 \quad 4 \quad 6 \quad 5 \right)$$

which differ only in the positions of c_9 and c_{10}. The first one gives a shelf layout of:

c_8	c_8	c_7	c_9	c_9	c_9	c_{10}	c_{10}	c_{10}	c_2	c_3	c_1	c_1	c_4	c_4	c_6	c_6	c_6	c_5	c_5

In this case, to keep categories c_7 and c_8 next to the header we had to increase their affinity to it, because keeping it as in the previous experiment did not achieve the desired result. This shows the sensitivity of the results to the selection of the affinity values.

3.2 Two shelves separated by an aisle - P2SAP2

We will consider now the case of two face-to-face shelves consisting of 10 and 13 modules of length 1 and separated by an aisle of width 3. There is also a reference point located at the left end of the aisle.

The chromosome is now a vector consisting of two parts. The first $S - 1$ components indicate the number of categories to be placed in a shelf (with S equal to the number of shelves). The remaining components indicate the order in which the categories will be placed in the shelves. Hence the length of the chromosome will be $S - 1 + K$.

For instance, let us assume $S = 2$ and $K = 8$. A chromosome such as:

$$\mathbf{chrom} = \left(\mathbf{3} \quad 8 \quad 5 \quad 1 \quad 3 \quad 2 \quad 6 \quad 4 \quad 7 \right)$$

indicates that there will be 3 categories in the first shelf (c_8, c_5 and c_1) and the remaining 5 (c_3, c_2, c_6, c_4, and c_7) will be placed in the second shelf. The vector $\mathbf{h}$ indicating the number of modules per category will be determined independently for each shelf following the highest averages method. We will also calculate a vector $\mathbf{h}_g$, which indicates the number of modules that would be allocated per category if all the modules were grouped in a single shelf, i.e., the *global* $\mathbf{h}$.

We will assume that:

- A category cannot be distributed over two shelves
- It is preferable to place two affine categories on the same shelf rather than split into two shelves, even when the Euclidean distance between modules of different shelves is less than between modules of the same shelf.

After trying different variations of the fitness function, we settled for:

$$f = \frac{1}{2}\sum_{i=1}^{K}\sum_{j=1}^{K}\left((d_{i,j}\cdot|a_{i,j}|)^{sgn(a_{i,j})}\cdot p_{i,j}\right)+\beta\cdot||\mathbf{h}_g-\mathbf{h}||_2 \quad (7)$$

where

$d_{i,j}$, $a_{i,j}$ and $sgn(a_{i,j})$ are as defined for Equation 3
the affinity matrix $\mathbf{A}$ is as for Equation 6
$||\mathbf{h}_g - \mathbf{h}||_2$ is the quadratic distance between $\mathbf{h}$ and $\mathbf{h}_g$

and the *penalty*, $p_{i,j}$, is defined as follows

$$p_{i,j} = \begin{cases} P & \begin{cases}\text{if } c_i \text{ and } c_j \text{ are on different shelves and } a_{i,j} > 0\\ \text{or if } c_i \text{ and } c_j \text{ are on the same shelf}\\ \text{and } a_{i,j} < 0\end{cases}\\ 1 & \text{otherwise}\end{cases}$$

Different values of P and β were tried and finally we settled for $P = 100$ and $\beta = 50$. We run the algorithm 30 times, with a population size of 200 and a time limit of 50 seconds per run, obtaining the following best solutions

$$\begin{aligned}\mathbf{chrom} &= \begin{pmatrix} 2 & 5 & 6 & 7 & 8 & 2 & 3 & 1 & 4 \end{pmatrix}\\ \mathbf{chrom} &= \begin{pmatrix} 2 & 5 & 6 & 8 & 7 & 2 & 3 & 1 & 4 \end{pmatrix}\\ \mathbf{chrom} &= \begin{pmatrix} 2 & 6 & 5 & 7 & 8 & 2 & 3 & 1 & 4 \end{pmatrix}\\ \mathbf{chrom} &= \begin{pmatrix} 2 & 6 & 5 & 8 & 7 & 2 & 3 & 1 & 4 \end{pmatrix}\end{aligned}$$

with a value of $\mathbf{h}$ as follows:

$$\mathbf{h} = \begin{pmatrix} 3 & 1 & 1 & 3 & 5 & 5 & 2 & 3 \end{pmatrix}$$

while the value of $\mathbf{h}_g$ is:

$$\mathbf{h}_g = \begin{pmatrix} 3 & 1 & 2 & 3 & 4 & 4 & 3 & 3 \end{pmatrix}$$

which differs from $\mathbf{h}$ in four values. The actual shelf layout for the first result is:

c_5	c_5	c_5	c_5	c_5	c_6	c_6	c_6	c_6	c_6

AISLE

c_7	c_7	c_8	c_8	c_8	c_2	c_3	c_1	c_1	c_1	c_4	c_4	c_4

which achieves the objectives of

- placing c_7 and c_8 close to the reference point (located in $(0, 0)$, to the left of the shelves),
- all categories belonging to the same group are together and on the same shelf
- the allocation of modules is relatively close to the "ideal" obtained by the highest averages method

3.3 A more complex topology - P2SAPn

We will address here the problem with a more complex topology: eight shelves in different positions, as shown in Figure 2. The approach we will take is to concentrate on the placing of *groups* instead of categories. This brings about two simplifications, namely:

- The components of the affinity matrix can take one of three values: -1, 0 and 1.
- A group can be distributed over more than one shelf.

As a consequence of the above, the matrices $\mathbf{A}$, $\mathbf{SS}$ and $\mathbf{D}$ now refer to groups and not categories. So, $a_{i,j}$ and $d_{i,j}$ are the affinity and distance, respectively, between groups g_i and g_j. And the columns of $\mathbf{SS}$ contain the standard, minimum and maximum number of modules per group.

The distance between groups is going to be taken now as the distance between their *centres of gravity.* We define the centre of gravity (c.o.g.) of a group as the module in the group for which the sum of the distances of itself to every other module in the group is minimal; if several modules fulfill this condition then the first one encountered will be taken as the c.o.g.

Another major difference between this and previous versions of the problem is that now the allocation of number of modules per group is going to be done as part of the evolutionary algorithm, rather than using a separate algorithm to calculate it. Also, because a group can be spread on different shelves, we must ensure that the group maintains sufficient cohesion, in other words, we have to control the *group dispersion.*

We will introduce a different encoding for the chromosome. Instead of a vector, we will employ a MATLAB cell array structure, which is a matrix in which the rows can different lengths. In our case each row i stores the modules allocated to group g_i. The restrictions are, as before, that the total number of elements must equal the total number of modules in the shop and that each chromosome represents a permutation of the modules' indices.

For instance, for $G = 3$ and $M = 10$ a possible chromosome would look like this:

$$\mathbf{chrom} = \begin{pmatrix} 9 & 7 & & & \\ 10 & 8 & 5 & 1 & 3\\ 2 & 6 & 4 & & \end{pmatrix}$$

meaning group 1 gets modules 9 and 7, group 2 gets modules 1, 3, 5, 8 and 10, and so on.

When defining the fitness function we must take into account three different objectives:

- group affinity,
- deviation from the standard shop in terms of number of modules, and
- dispersion of the modules belonging to a group.

The last two objectives are specific of this version of the problem. The dispersion refers to how far apart the modules allocated to a group are located. To measure this, we introduce two penalty coefficients in the evaluation function:

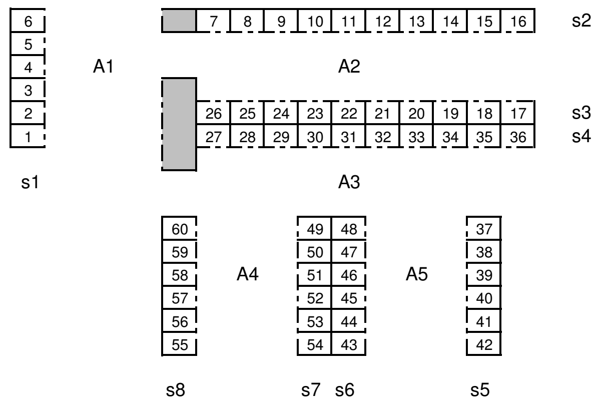

Figure 2: Shop layout for P2SAPn, with 5 aisles, 8 shelves and 60 modules; the numbers represent the module identifiers, m_i. The dashed lines represent the accessible side of the shelf; the shaded areas represent shelf headers, used as reference points.

- *Inter*-group penalty, P: to penalise the situation in which modules belonging to the same group are located in different corridors, and

- *Intra*-group penalty, δ: to penalise the dispersion of the modules belonging to a group.

Notice how for this problem the module allocation vector **h** is given by the evolutionary algorithm and not calculated beforehand, as in previous Sections. The resulting fitness function is as follows:

$$f = \frac{1}{2}\sum_{i=1}^{g}\sum_{j=1}^{g}\left((d_{i,j}\cdot|a_{i,j}|)^{sgn(a_{i,j})}\cdot p_{i,j}\right) + \\ + \sum_{r=1}^{g} disp_{g_r} + ||\mathbf{std}-\mathbf{h}||_2 \quad (8)$$

where:
$d_{i,j}$ is the distance between the *centres of gravity* of groups g_i and g_j;
$a_{i,j}$ is the affinity between groups g_i and g_j;

$$p_{g_i,g_j} = \begin{cases} P & \begin{cases}\text{if } c.o.g._{g_i} \text{ and } c.o.g._{g_j} \text{ are on different} \\ \text{aisles and } a_{i,j} > 0 \\ \text{or if } c.o.g._{g_i} \text{ and } c.o.g._{g_j} \text{ are} \\ \text{on the same aisle and } a_{i,j} < 0\end{cases} \\ 1 & \text{otherwise}\end{cases}$$

$||\mathbf{std}-\mathbf{h}||_2$ measures the difference between the number of modules per group assigned by the algorithm and that given by the first column of the standard shop matrix **SS**. It is calculated as a quadratic distance between vectors;
$\sum_{i=1}^{G} disp_{g_i}$ is the total dispersion of the modules within all groups, where $disp_{g_i}$ is calculated as follows:

$$disp_{g_i} = \frac{\sum_{k=1}^{M_{g_i}}(cog_{g_i}, m_k)}{\frac{M_{g_i}^3}{12}} + S_{g_i} + \delta \quad (9)$$

S_{g_i} is the number of different shelves occupied by g_i
M_{g_i} is the number of modules allocated to g_i
δ penalizes disperse modules as follows:

$$\delta = \begin{cases} R & \text{if all the group modules are not placed in the same aisle,} \\ 0 & \text{otherwise}\end{cases}$$

Finally, evolutionary operators have also been modified to accommodate the new representation. The following mutation operators were implemented:

- `Number of modules mutation`: the number of modules of a group is increased or decreased

- `Modules mutation`: two groups exchange a random number of modules

- `Groups mutation`: all the modules corresponding to two groups are exchanged

All other parameters of the evolutionary algorithm remained the same.

For the experiments we defined six groups, g_1 to g_6, and two reference points (located at the headers of the shelves), which were modeled as two extra groups, g_7 and g_8. The affinity matrix (now storing group affinity values) was as follows:

$$\mathbf{A} = \begin{pmatrix} - & 0 & 0 & -1 & 0 & 0 & 1 & 1 \\ 0 & - & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & - & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & - & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & - & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & - & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & - & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & - \end{pmatrix} \quad (10)$$

In Equation 10 it can be seen that g_1 is affine with groups g_7 and g_8, i.e., it is preferable to place this group in the shelves closer to the reference points.

The standard shop requirements (now referring to group standards rather than the category standards) employed in the experiments are given by Equation 11, where the last column of the matrix indicates that all the groups have the same preference.

$$\mathbf{SS} = \begin{pmatrix} 6 & 5 & 9 & 1 \\ 6 & 6 & 6 & 1 \\ 6 & 4 & 8 & 1 \\ 12 & 5 & 15 & 1 \\ 10 & 7 & 13 & 1 \\ 19 & 10 & 20 & 1 \end{pmatrix} \quad (11)$$

We ran the algorithm 50 times, with a termination criterion of 30,000 iterations (approximately 25 minutes per run). Most runs (78%) obtained good solutions; seven of them (14%) obtained the best chromosome:

$$\begin{pmatrix} 1\ 2\ 3\ 4\ 5\ 6 \\ 27\ 28\ 29\ 30\ 31\ 32 \\ 33\ 34\ 35\ 36 \\ 37\ 38\ 39\ 40\ 41\ 42\ 43\ 44\ 45\ 46\ 47\ 48 \\ 49\ 50\ 51\ 52\ 53\ 54\ 55\ 56\ 57\ 58\ 59\ 60 \\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17\ 18\ 19\ 20\ 21\ 22\ 23\ 24\ 25\ 26 \end{pmatrix}$$

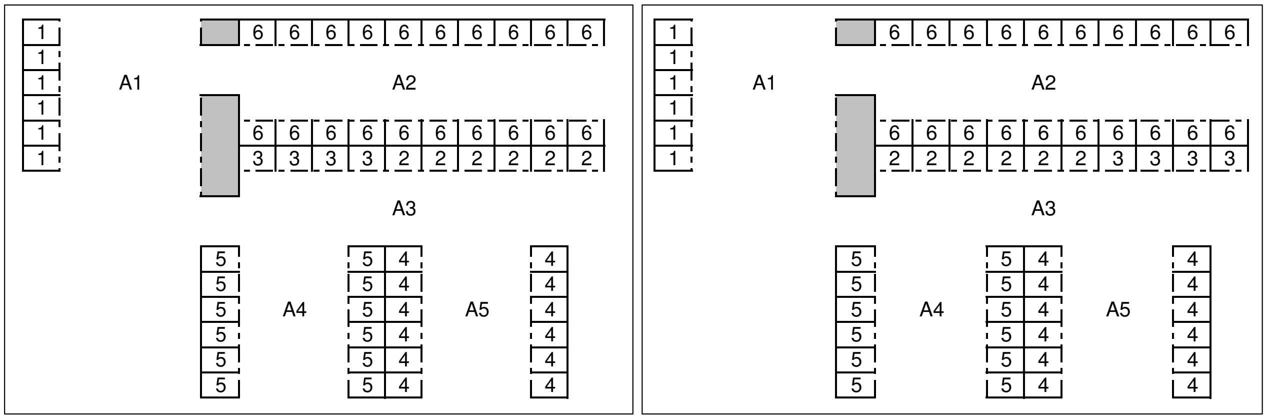

Figure 3: Allocation of groups to modules for two solutions of equal fitness

depicted in Figure 3, right, or a similar solution of equal fitness, depicted in Figure 3, left, in which groups 2 and 3 swap positions:

$$\left(\begin{array}{l}
1\ 2\ 3\ 4\ 5\ 6 \\
31\ 32\ 33\ 34\ 35\ 36 \\
27\ 28\ 29\ 30 \\
37\ 38\ 39\ 40\ 41\ 42\ 43\ 44\ 45\ 46\ 47\ 48 \\
49\ 50\ 51\ 52\ 53\ 54\ 55\ 56\ 57\ 58\ 59\ 60 \\
7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17\ 18\ 19\ 20\ 21\ 22\ 23\ 24\ 25\ 26
\end{array}\right)$$

For both solutions the numbers of modules assigned to each group were:

$$\mathbf{h} = (\ 6 \quad 6 \quad 4 \quad 12 \quad 12 \quad 20\ )$$

which is "close enough" to the group standards, expressed in the first column of Equation 11:

$$\mathbf{std'} = (\ 6 \quad 6 \quad 6 \quad 12 \quad 10 \quad 19\ )$$

Notice the contribution to the result of the three components of the fitness function. First, the restrictions imposed by group affinities are accomplished: group 1 is close to the reference points, groups 2 and 3 are close to each other and groups 1 and 4 apart from each other. Second, all the modules belonging to the same group are placed together (thus minimising the dispersion). And finally, the number of modules assigned to each group is similar to the standard values.

4. CONCLUSIONS AND FURTHER WORK

We have shown that our evolutionary algorithm can be used as a time-efficient tool to obtain solutions for the P2SAP.

Future work will involve:

- More complex topologies with higher number of shelves and more reference points.
- Studying the contribution of the different components of the fitness function; also considering the possibility of addressing the problem as multi-objective.
- Selecting a unique representation that can be applied to all versions of the problem. Also, studying how to address it with Genetic Programming rather than a Genetic Algorithm.
- Employing an efficient algorithm to calculate the affinity matrix. Associating complementary products, because costumers need them at the same time to achieve a specific goal, is the most commonly employed method [3][8][11][12].

5. ACKNOWLEDGEMENT

This work was part of the *ANaLog* project and is supported by the Instituto de la Pequeña y Mediana Industria Valenciana (IMPIVA), N^o Exp. IMCITA/2005/35, and partially funded by the European Regional Development Fund.

6. REFERENCES

[1] Anderson, E.E., Amato, H.N.: A mathematical model for simultaneously determining the optimal brand collection and display area allocation, *Operations Research*, 22: 13-21 (1974)

[2] Bai, R. and Kendall, G.: An investigation of automated planograms using a simulated annealing based hyper-heuristic. Automated Scheduling, Optimisation and Planning Research Group, School of Comp. Sci. and IT, Univ. of Nottingham.

[3] Brijs, T., Swinnen, G., Vanhoof K. and Wets, G. Using association rules for product assortment decisions: a case study. *Procs. 5th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, San Diego, USA, 254-260 (1999)

[4] Corbett R., Jacobs F., Shackleton, M.: The European Parliament. *London: John Harper* (2001)

[5] Corstjens, M., Doyle, P.: A model for optimizing retail space allocation, *Management Science*, 27(7): 822-833 (1981)

[6] Eiben, A. E., Hinterding, R. and Michalewicz, Z.: Parameter control in Evolutionary Algorithms. *IEEE Trans Evol Comp*, 3: 124-141 (1999)

[7] Hansen, P, Heinsbroek, H: Product selection and space allocation in supermarkets. *Eur J Oper Res*, 3:474-484 (1979)

[8] Henderson, James B., Quandt, R.: Micro Economics Theory: a mathematical approach. *New York: McGraw-Hill* (1980)

[9] Lim, A., Rodrigues, B. and Zhang, X.: Metaheuristics with local search techniques for retail shelf-space optimization. *Management Science*, 50(1):117-131 (2004)

[10] Palomares, R.: *Merchandising, cómo vender más en establecimientos comerciales.* Gestión 2000, Barcelona. Pp. 127- (2001)

[11] Shahidi, A.: The End of Supermarket Lethargy: Awakened Consumers and Select Innovators to Spur Change. *Supermarket Industry Perspective* (2002)

[12] Walters, R.G.: Assessing the impact of retail price promotions on products substitution, complementary purchase, and interstore sales displacement. *J. Marketing*, 55(1):17-28 (1991)