ADDRESSING THE FACILITIES LAYOUT DESIGN PROBLEM THROUGH CONSTRAINT LOGIC PROGRAMMING

José Tavares[1], *Carlos Ramos*[1] & *José Neves*[2]

[1]Dept. of Computer Engineering – Polytechnic Institute of Porto (IPP)/Institute of Engineering
Rua Dr. António Bernardino de Almeida, 4200-072 Porto – Portugal
Phone: +351 2 8340500, Fax: +351 2 8321159, Email: {jtavares,csr}@dei.isep.ipp.pt

[2]Dept. of Informatics – University of Minho
Campus de Gualtar - 4709 Braga Codex – Portugal
Phone +351 53 604466, Fax: +351 53 604471, Email: jneves@di.uminho.pt

Abstract

One of the most difficult problems that face researchers experimenting with complex systems in real world applications is the Facility Layout Design Problem. It relies with the design and location of production lines, machinery and equipment, inventory storage and shipping facilities. In this work it is intended to address this problem through the use of Constraint Logic Programming (CLP) technology. The use of Genetic Algorithms (GA) as optimisation technique in CLP environment is also an issue addressed. The approach aims the implementation of genetic algorithm operators following the CLP paradigm.

Keywords: Plant Layout, Facilities Layout, Constraint Satisfaction, Constraint Logic Programming, Layout Design.

1. Introduction

1.1 The Problem

The Facility Layout Design Problem (FLDP) is one of the most complex industrial problems. It looks for an efficient physical arrangement of machines, cells or departments, which are collectively named as facilities. Methods to solve these problems have to deal with a large set of factors, namely sales and production estimation, manufacturing process compatibilities, delivery dates, quality, spatial requirements, economics, management, human resources and environment.

In a more general definition, the FLDP is the planning of the proper location of machines, employees, workstations, warehouses and client service areas. It also involves the design of the material and people flow pattern around, the movement inside, at the input and at the output of the productive plants. In a factory, the layout is a fundamental issue. From it, the equipment and human resources have a great influence on the real output, whatever is the manufacturing plant's theoretical installed capacity. It is necessary to plan the operations scheduling among the available equipment for each operation type and the flow of the materials and people among them. The warehouses location, how they are supplied from outside, the areas and how the distribution transportation are loaded are also tasks of the planning process. Issues related with layout, like work conditions (noise levels, temperature and air quality), have to be considered. The correct design and the dynamic management of the manufacturing plant is a manager's fundamental task in order to have an efficient manufacturing process using the available material and human resources.

The FLDP was originally defined by /1/ and /2/. Given the complexity of the FLDP, a strong effort was given in the

research and development of techniques, which aims to help the specialist to solve it /3/, /4/, /5/, /6/, /7/. These techniques use procedures classified as optimal and sub optimal algorithms. For the first ones, the attainment of the optimal solution for problems with some dimension has shown problematic and, therefore, other ways were explored giving good solutions in useful time. These algorithms are in the group of the sub optimal algorithms. All these techniques are usually based on Operational Research (OR) models and are usually classified into two types, the single-row layout and multi-row layout problems. As the name indicates, in the single-row layout problem the facilities are arranged linearly in one row as opposed to multi-row layout problem, where the facilities are arranged in two or more rows. One classical example of the multi-row layout problem type is the *Quadratic Assignment Problem* (QAP), /3/, /5/, /7/ which assumes that the manufacturing plant is divided into *n* equal areas, where the facilities are located. The cost function usually considers the distance and the flow between facilities.

Since FLDP is a complex problem (the simple QAP is NP-hard /3/, /7/), optimal algorithms were not good enough for large and real problems. Examples of those optimal algorithms are the Branch-and-Bound algorithm, the decomposition algorithm and the cutting plane algorithm /7/. In practice, heuristic based algorithms (sub optimal algorithms) are used to find one good solution /6/, which are classified as construction algorithms, improvement algorithms, hybrid algorithms and graph theoretic algorithms. The construction algorithms generate a facility layout from scratch, which means that a layout is built in a single iteration (the ALDEP and CORELAP are two examples of these algorithms /3/, /5/, /7/). The improvement algorithms require an initial layout, and then several operations are applied in order get solution improvements (an example of the improvement algorithms is the CRAFT algorithm /3/, /5/, /7/). Hybrid algorithms are the ones that use two or more types of techniques or the ones that use a combination of optimal algorithms with heuristics. Finally, the graph theoretic algorithms are based on the graph theory, namely planar graph and maximal planar graph concept /7/.

Meta-heuristic algorithms like Simulating Annealing, Taboo Search and Evolutionary Algorithms have been used also to solve the FLDP. An approach found frequently in the literature is the optimisation with evolutionary computation techniques. A survey about the use of these techniques to solve the FDLP can be found in /8/.

In the modern manufacturing systems, the traditional FDLP assumptions are more and more difficult to support. In first place, there is a tendency to consider a third dimension given, for example, lighter machines, higher prices of the available areas, among others. In second, it is evidenced that in the current industrial environment, there is a strong trend for an increasing level of volatility and uncertainty, where more and more companies are present in a global market. It is also evidenced, an increasing technological innovation and changes in the specifications of the products, these demanded by the consumers. All these factors contribute to reduce the life cycle of a manufacturing layout and, thus, an increasing need of better computational tools to help the layout designer to create new manufacturing layouts or the re-layout of the old ones.

1.2 The Technology

In the last decade a new technology has emerged to deal with complex combinatorial problems. This technology is known as Constraint Logic Programming (CLP) /9/ and matches the declarative aspects of the Logic Programming (LP) paradigm with the techniques for constraint satisfaction /10/, in a proper way for problem solving. This hybrid technique improves the search strategies used in logic programming, once it adds constraints and consistence verification techniques. With this scheme, the solution space can be largely reduced.

The constraints and consistency verification techniques were initially developed to solve the Constraint Satisfaction Problems (CSP), which for a long time had been an Artificial Intelligence (AI) research field. Many combinatorial problems, characterized by a large number of constraints, are well suited for CLP, namely scheduling problems, timetabling, planning, placement, configuration, and routing. Others areas of application goes from the natural language

processing, to the circuit analysis and games theory. CSP seeks assignments to a set of variables $X = \{x_1, x_2, ..., x_m\}$ from a set of corresponding domains $D=\{d_1, d_2, ..., dm\}$, one per variable, satisfying a set of constraints $C=\{c_1, c_2,..., c_n\}$ over subsets of the cartesian space spanned by *D*. CSP is a binary problem, in which a set of assignments to the variables *X* satisfies or not all the constraints /9/, /10/, /11/. A solution for a CSP is a domain value assignment for each variable, in a way that all the constraints are satisfied. It has been verified that CLP offers a more natural way to express real world problems in a computer program, the development time is shorter, the maintenance processes are simpler and the efficiency is equivalent to that of the programs developed in procedural languages according to the paradigm of constraint satisfaction /11/.

Since the end of the eighties the CLP technology, and in particular the Constraint Logic Programming with Finite Domains (CLP(*FD*)) /9/, /11/, has been applied to solve problems, with a great success, in several areas where other technologies had lapsed. In relation with the industrial applications, the production planning and scheduling have been the elected areas. Many of these problems present common features to the combinatorial problems and, therefore, they are difficult to solve. As it was referred, the FDLP are also complex problems and, therefore, solving them is hard. Given the complexity of the FLDP and the considerable amount of work that has been done in FLDP area over the last three decades we intended to contribute with a work which explores the CLP(*FD*) technology to solve this kind of problems.

Solving the FDLP with the CLP(*FD*) technology requires, however, the development of new models or, at least, the adaptation of some models already used with other technologies. One fundamental component of this document is to describe a formal model to solve industrial FDLP, emphasising the aspects related with the use of the CLP(*FD*) technology to solve it. This model was inspired in models of space assignment problems /12/, /4/, /13/. Another fundamental component is related with the identification of the problem variables as well as with the definition of its domains and, basically, with the specification of the constraints, that obviously have a geometric nature.

According to the results obtained by applying CLP technology to solve complex combinatorial problems, an early approach that uses CLP for solving the FLDP was developed /14/, /15/, /16/, /17/, /18/. However, it was verified with this approach that the optimisation task, which uses a Branch&Bound (*B&B*) algorithm offered by the main CLP development tools, requires a huge computational power to explore the entire search space for real problems of this kind. This scenario suggests that other optimisation techniques should be used in order to deal with such huge search space. The chosen technique was the Genetic Algorithms (GA) /23/, /24/, which are general-purpose search procedures based on natural selection and evolutionary principles /25/. The approach followed is a combination of CLP and GA, which is presented in this paper. We claim that this combination is in fact better than the use of CLP alone with the build-in *B&B* algorithm.

2. Information Requirements for FDLP

In this section we identify the required input data for the model we propose. This model was developed having in mind that we intended to solve problems using CLP(*FD*) solvers. However, we start by introducing some general concepts related with the FDLP.

2.1 FDLP Models

Globally, all the models used to solve this kind of problems are complex to handle. In geometric terms, we are dealing with facilities requiring a fraction of the available space in the manufacturing plant. We refer to the manufacturing plant as the available space to place facilities, usually a building or some part of it. In general, the space requirements of facilities to place in the plant can be grouped in: (i) equal area and fixed orientation; (ii) different areas and fixed orientation; (iii) different areas and variable orientation; (iv) different areas and variable shapes. In the approach (i) and (ii) it is necessary to choose the location for each facility. In (iii) a new dimension to the complexity of the problem is

added since it is also necessary to choose the orientation for each facility in the plant. Finally, in (iv), the dimension related with the orientation is replaced with another that implies the selection of the facility shape. Selecting the facility shape is equivalent to chose the width and length values since we are dealing with rectangular shapes.

This FDLP models classification based on space requirements assumes manufacturing plants with only one floor. However, it has been proposed variants of these models to deal with several floors, which, obviously, adds another dimension to the problem, increasing the complexity. However, in this work we are just considering manufacturing plants of one floor.

Besides the geometric factors, it is still necessary to consider the factors related with the productive process. The productive process is chosen based on the products to manufacture and the productive capacity to install in order to satisfy the product demand. In general, the product demand is directly or indirectly estimated. The product demand volume is one parameter required to evaluate and choose the best solution for each instance of the problem.

2.2 Manufacturing Plant

The knowledge of the plant dimensions is not, in almost all situations, essential to find the best solution, when the available physical space in the plant is not a constraint. Moreover, this knowledge of the plant length and width may help finding the best solution. Also, given the plant shape, the available physical space may not be compatible with the best solution found. Therefore, it is desirable that the plant dimensions should be took into account.

The problem solutions taking into account the plant dimensions undertake implicitly a manufacturing facility whose plant has a rectangular shape with length L and width W. It is obvious that not all the plants have a rectangular shape. In those cases the shape considered is a rectangle surrounding the real plant (Fig 1). This approach has a disadvantage to potentially give rise to solutions that locates some facilities in a way that crosses the real plant frontiers. To avoid this shortcoming, our solution places constraints that exclude the areas that do not belong to the real plant. This type of constraints will be presented with more detail below. Also, this approach can also exclude interior areas of the plant that are not available to place the facilities. Taking into account what was discussed about the plant shape, the representation has consider the following:

- W is the width of the rectangle surrounding the plant shape;
- L is the length of the rectangle surrounding the plant shape;
- NAP is the number of the forbidden areas;
- $\{AP_i\}$ is the set of the forbidden areas.

By convention the width W is measured in the x-axis while the length L is measured in the y-axis.

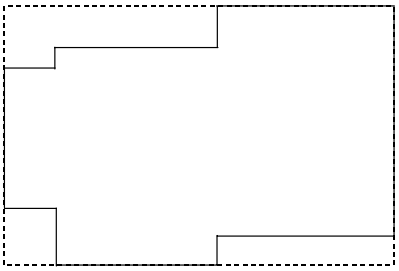


Fig 1: An example of a rectangle surrounding the manufacturing plant.

The forbidden areas are rectangular and are described in terms of the following parameters:

- X_i is the value of the x co-ordinate of the area i ;
- Y_i is the value of the y co-ordinate of the area i ;
- W_i is the width of the rectangular area i ;
- L_i is the length of the rectangular area i .

2.3 Facilities

Facilities in the FDLP context are plant spaces used for the most varied purposes, as for example, the ones for services, productive warehouses and/or processes. In this work we are interested with facilities related with the productive process. These facilities could be a simple workstation with a machine and, optionally, with a small area for temporary storage of materials, or a collection of workstations where the facility itself is a layout sub problem.

In general, the facilities where the process operations

occur are known. In the model that we are describing we also acknowledge that there is some alternative facilities to perform the same process operation. The set of facilities that are able carry out the same process operation we call a facility class.

Each facility is identified by a set of properties that are related with its shape. There are also other properties, which are related with the facilities capacity to accomplish the operations. However, these are directly related with the productive process. The description of a facility has, therefore, to take into account that:

T_i is the facility class *i ;*

NI_i is the number of facilities of class *i* ;

I_{iu} is the facility *u* of class *i* ;

A_{iu} represents the minimal area required by I_{iu} ;

$\{AR_{iu}\}$ is set of values that represents the possible aspect ratios of I_{iu};

W_{iu} is the width of I_{iu} ;

L_{iu} is the length of I_{iu} ;

G_{iu} is an optional gap value of I_{iu} and represents the minimal distance that has to be respected in relation to the others facilities (Fig 2).

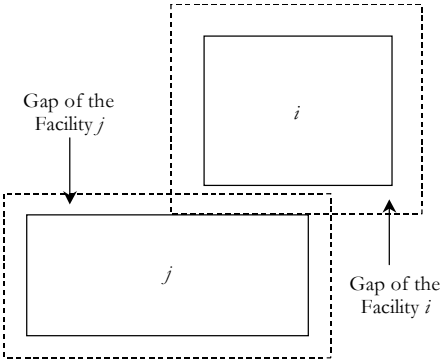


Fig 2: The gap between two facilities.

The width and length as well as set of values of the Aspect Ratio (AR) are equivalents, being therefore redundant. Expressions (1) and (2) allows to relate these values.

$$L_{iu} = \sqrt{AR_{iu} \times A_{iu}} \quad (1)$$

$$A_{iu} = W_{iu} \times L_{iu} \quad (2)$$

It is clear that if we have several possible values of AR then there is also several values for *W* and *L*. On the other hand, when a facility requires a fixed area, it is only enough to know the values of *C* or *L*, once (2) relates each other. In general the value of the required area and the set of the AR values are enough to treat all the situations related with facilities shape. Tab 1 shows the three major cases that may occur.

Tab 1: The three major cases for the shape of a facility.

Facility Shape	**Possibilities**	***AR***
Fixed orientation	1	$\{v\}$
Variable orientation	2	$\{v, \frac{1}{v}\}$
Variable shape	*N*	$\{v_1, ..., v_n\}$

2.4 Products

One company exists since there is a market wishing to consume a large and diverse number of products, being the company able to satisfy some or all the market needs in a certain niche of products. In this section it is showed how a company could see the demand of the market for products manufactured in the layout point of view and how this demand affects the process that occurs in the plant that wishes to design.

The knowledge of the products to be produced in the new plant is essential in the choice of the manufacturing process. However, it is the foreseen volume of products to manufacture that imposes the capacity of the plant, conditioning the decisions that make the plant efficient in the production, namely in the choice of the best disposal for the facilities inside the plant. The choice of the best disposal for facilities depends essentially on the flow of materials or on the frequency of trips of the carrier equipment between the facilities.

To compute the flow between facilities it is necessary to decompose the products in their parts. It is obvious that this decomposition is restricted to the parts that are processed in

the plant. This decomposition is done in accordance with the Material Requirements Plan (MRP). For each part the required amount must be computed (the demand value of the part). Collectively, the parts and the final products are treated simply as parts. The information that describes those parts is given in the form:

NP is the number of parts in the manufactured plant;

P_k is the part k ;

C_k is the manufacturing capacity of the part k;

O_{ikl} is the order number of the operation l that is done in the facility class i, to the part k.

2.5 Production Process

Knowing what parts are going to be manufactured in the plant, and the amount of each part per unit of time, it is necessary to know the operations sequence to compute the flow of materials between the facilities. To do this, we firstly have to decompose all the products in their simpler parts. After this decomposition we know the sequence of operations of each part and, therefore, the routing between facilities.

To specify the sequence of operations, taking into account the product, we have the following:

NO_k is the number of operations applied to part k ;

NO_{ik} is the number of operations applied to part k in the facility class i ;

O_{ikl} is the order number of the operation l done in facility class i, to the part k, in the sequence of operations;

$\{C_{kiu}\}$ is a list containing values representing the number of parts k processed in the facility u, of the class i, for unit of time;

L_k is the transportation lot size of the part k ;

T_k is the transportation cost for each part k .

With this data it is possible to compute the flow between all the facilities pairs. The expression (3) allows the computation of the flow of the part k between the facilities of class i and j. Notice that this flow value is not zero only when the two operations involved are consecutive. The total flow value between the facilities of class i and j, for all parts, is given by the expression (4).

The flow value computed with (3) and (4) is the flow between facilities classes and not between instances of the facilities classes. In most cases it is not possible to know in advance, during the layout planning, the instance of each facility class that is used to perform the operations. In this way, it was stipulated that the flow that leaves and arrives to the facilities of the same class is proportional to the amount of processed parts for each facility and for a unit of time.

$$F_{ij}^{k} = \begin{cases} \dfrac{T_k \times C_k}{L_k} & if \quad \left|O_{ikl} - O_{jkw}\right| = 1 \\ 0 & otherwise \end{cases} \tag{3}$$

$$f_{ij}^{1} = \sum_{k=1}^{NP} F_{ij}^{k} \tag{4}$$

The flow between facility u of class i and facility v of class j, related with part k, is computed according to expression (5), C_{ik} and C_{jk} are given by expression (6). The C_{ik} and C_{jk} values represent the total number of parts k processed in all the facilities of class i and j, respectively. The total among of flow of all parts between facility u of class i and facility v of class j, is then computed by (7).

$$F_{iujv}^{k} = F_{ij}^{k} \times \frac{C_{iuk}}{C_{ik}} \times \frac{C_{jvk}}{C_{jk}} \tag{5}$$

$$C_{ik} = \sum_{u=1}^{NI_i} C_{iuk} \tag{6}$$

$$f_{iujv}^{1} = \sum_{k=1}^{NP} F_{iujv}^{k} \tag{7}$$

The flow values computed until the moment are only related with the transport of materials between facilities for operations performed to the same part. The transportation of subparts to a facility that performs an operation that groups subparts in a more complex part is not treated directly. When the cost associated with the transport to perform assembly operations is not negligible it is necessary to consider additional information that deals with the flow due to incorporation subparts in a more complex part. This additional information is given in the form:

k is the complex part k incorporating several subparts;

NSP_k is the number of subparts that needed to the part k;

$\{P_l\}$ is the list of subparts needed to assembly the part k ;

$\{q_{kl}\}$ is a list of values, being each one the quantity of subparts l needed to assembly the part k.

With this information, the computation of flow resulted by the incorporation in a complex part of several subparts, is carried using the expression (8). The total flow, of the incorporation of all subparts in all the parts, can then be computed by the expression (9). The set of all the values of flow between all the pairs of facilities, allows us to build the flow array (10).

$$F_{iujv}^{kl} = \begin{cases} \frac{T_l \times C_k \times q_l}{L_l} \times \frac{C_{iuk}}{C_{ik}} \times \frac{C_{jvk}}{C_{jk}} & if \quad O_{iku} = 1 \quad e \quad O_{jkv} = NO_l \\ 0 & otherwise \end{cases} \tag{8}$$

$$f_{iujv}^{2} = \sum_{k=1}^{NP} \sum_{l \in \{P_l\}}^{NSP_k} F_{iujv}^{kl} \tag{9}$$

$$f_{iujv}^{M} = f_{iujv}^{1} + f_{iujv}^{2} \tag{10}$$

Taking into account the described method to compute the flow between all the facility pairs, the cost of a layout solution can be computed with the expression (11), where d_{ij} is in the distance between facilities i and j, being this usually given, by an euclidean or a rectilinear metric. This distance is dependent, obviously, of the position where the facilities are placed.

$$Custo = \sum_{j=1}^{n-1} \sum_{i=j+1}^{n} \sum_{u=1}^{NI_i} \sum_{v=1}^{NI_j} f_{iujv}^{M} \times d_{iujv} \tag{11}$$

3. FDLP Modelling with CLP(*FD*)

After establishing the information requirements to solve the FDLP and the way how the plant, the facilities and the processes are modelled, in this section we define how this modelling can be addressed using CLP(*FD*).

3.1 Variables

Solving a FDLP using CLP(*FD*) involves the selection of the best location to place the facilities inside the plant. It also involves the selection of the best shape for the facilities. It is assumed that facilities shape is considered as being rectangular. As consequence, solving an FDLP requires four decision variables for each facility, two for the facilities coordinates and two for their shape. In relation to the facilities coordinates, the associated variables points to their geometric centre.

CLP(*FD*) solvers use decision variables that can take only values in a subset of integer numbers. However, the information requirements for solving FDLP suggest that decision variables should take real values. It is important, therefore, to have into account this aspect, since it is necessary to make conversions from real values to integers values, with the consequent loss of accuracy. Depending on the wanted accuracy, some times it will be necessary to scale the values, before doing the conversion.

In relation to the coordinates of the installations, its domain has to contemplate the dimensions of the plant. In this way, the domain of the coordinates is restricted by the constraints (1) and (2):

$$X_{iu} \in 0 \,..\, W\text{-}1 \tag{1}$$

$$Y_{iu} \in 0 \,..\, L\text{-}1 \tag{2}$$

where:

X_{iu} and Y_{iu} Represents the x and y coordinate of the facility u of class i position;

W and L is the width and the length of the rectangle surrounding the plant shape;

Since the facilities central point matches with their coordinates in plant, only half of their width and the half of their length have to be known. Taking into account the information requirements, three situations can be identified:

1. the width and length of a facility is known and is enough to chose the best orientation;
2. it is specified a minimum area and an interval of AR values to the facility;
3. it is specified a minimum area and set of AR discrete values to the facility shape.

In the first situation the facility orientation in the plant is treated implicitly. The domain size of the width and length variables is two if their value is not equal and is one, and

therefore these variables get instantiated, if they are equal. In general the domain of these variables are specified with the constraints (3) and (4).

$$W_{iu} \in [w_{iu}, l_{iu}] \quad (3)$$

$$L_{iu} \in [w_{iu}, l_{iu}] \quad (4)$$

where

W_{iu} and L_{iu} is the domain variable related with the width and length of the facility u of class i ;

w_{iu} and l_{iu} is half of the width and half of the length values of the facility u of class i ;

If c_{iu} and l_{iu} values are not equal then the constraint (5) is added in order to avoid the facility shape is not going to be square.

$$C_{iu} \neq L_{iu} \quad (5)$$

In relation to the second situation, where we want to find which is the best shape to the facilities from a continuous interval of possible shapes, the constraints (6) and (7) are associated, respectively, to the decision variables related with the facilities width and length.

$$W_{iu} \in wi_{iu}..ws_{iu} \quad (6)$$

$$L_{iu} \in li_{iu}..ls_{iu} \quad (7)$$

where:

wi_{iu} and li_{iu} is the minimum value of the domain to the variable that represents half of the width and length, respectively, of the facility u of class i ;

ws_{iu} and ls_{iu} is the maximum value of the domain to the variable that represents half of the width and length, respectively, of the facility u of class i ;

Since the set of RA_{iu} values is an continuous interval, these four values ci_{iu} , li_{iu} , cs_{iu} and ls_{iu} , are given by the expressions (8), (9), (10) and (11), respectively.

$$li_{iu} = \left\lfloor \frac{\sqrt{\min\{AR_{iu}\} \times A_{iu}}}{2} \right\rfloor \quad (8)$$

$$ls_{iu} = \left\lceil \frac{\sqrt{\max\{AR_{iu}\} \times A_{iu}}}{2} \right\rceil \quad (9)$$

$$wi_{iu} = \left\lfloor \frac{\sqrt{A_{iu} \times 1/\max\{AR_{iu}\}}}{2} \right\rfloor \quad (10)$$

$$ws_{iu} = \left\lceil \frac{\sqrt{A_{iu} \times 1/\min\{AR_{iu}\}}}{2} \right\rceil \quad (11)$$

After specifying the domains of the variables L_{iu} and C_{iu} , it is also necessary take into account the following constraint:

$$4 \times W_{iu} \times L_{iu} = A_{iu} \quad (12)$$

which assures that the minimum area of the facility is maintained. Since in CLP(*FD*) it is only possible to instantiate variables with integer values, the number of possibilities for the shape of a facility is enumerable and is not infinite as given by AR continuous interval. This resulting number of possible shapes is not always enough. In the worst case it will be equivalent to first situation where we only are interested the facilities orientation. To get more shape possibilities we could affect all the geometric related variables by a scale factor before make rounding real values to integer values. Fig 3 shows four possible shapes as result from the domain specification of the width and length variables given an continuous interval of AR.

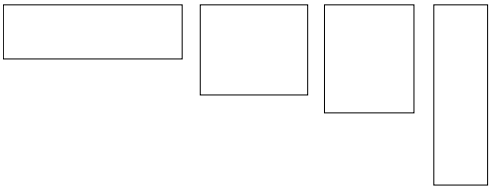


Fig 3: Four possibilities for the shape of a facility given a shape defined by a continuous interval of AR.

Finally, when the RA_{iu} values are supplied by a set of discrete values, the specification of the domains of W_{iu} and L_{iu} have to take into account each shape in the set. To do this, given the area and the set of AR for the facility shape, we build a set of possible widths ($\{w_{iu}\}$) and a set of possible lengths ($\{l_{iu}\}$) respecting the order of values in the set of AR. With these two new sets, the constraints (13) e (14) can be placed. They allow the establishment of a functional dependence between the width values and the length values of a facility by using a domain variable I_{iu}, and as such, the order of the values in the set is important.

$$element(I_{iu}, \{w_{iu}\}, W_{iu}) \quad (13)$$

$$element(I_{iu}, \{l_{iu}\}, L_{iu}) \quad (14)$$

Each pair of width-length values in $\{w_{iu}\}$ and $\{l_{iu}\}$ are computed as follow:

1. compute the values of the expressions (15), (16), (17) and (18) which denote the integer values (minimums and maximums) closer to the real values of the width and length of a facility;
2. for each possible combination given in 1 ($w \times l$), compute de area;
3. select the combination that gives the smaller area yet bigger than the minimum area required by the facility.

$$l_{iu}^{+} = \left\lceil \frac{\sqrt{AR_{iu} \times A_{iu}}}{2} \right\rceil \quad (15)$$

$$l_{iu}^{-} = \left\lfloor \frac{\sqrt{AR_{iu} \times A_{iu}}}{2} \right\rfloor \quad (16)$$

$$w_{iu}^{+} = \left\lceil \frac{\sqrt{A_{iu} \times 1/AR_{iu}}}{2} \right\rceil \quad (17)$$

$$w_{iu}^{-} = \left\lfloor \frac{\sqrt{A_{iu} \times 1/AR_{iu}}}{2} \right\rfloor \quad (18)$$

The domain of the variables, that defines the facilities coordinates, specified by (1) and (2), is not enough to place the facilities completely inside the plant. The constraints (19) to (22) have to be specified in order to get valid solutions.

$$X_{iu} \geq C_{iu} \quad (19)$$

$$X_{iu} + C_{iu} \leq C \quad (20)$$

$$Y_{iu} \geq L_{iu} \quad (21)$$

$$Y_{iu} + L_{iu} \leq L \quad (22)$$

3.2 Constraints

When solving the one FDLP, the facilities are placed in the plant in way that all constraints are satisfied. Constraints to avoid the overlapping of the facilities in the plant are always present. There is a second group of constraints that are used to guarantee the satisfaction on the solutions of the specific requirements for each instance of the problem. These specific requirements, between others, are usually technological, geometric, strategic and environment constraints, and should be indicated by layout designer to the system. It could also be pointed out a third group of constraints used to guide search of good solutions. These constraints could translate particularities of the problem and the experience of the experts. To deal with all these situations a set of constraint types was identified. These are:

1. **No Overlap** is the constraint that should always be present and which imposes that any facility must be placed in the plant in such way that is not going to overlap with the others;
2. **Neighbourhood** is used to deal with situations where it is desirable to locate two facilities close to each other as, for example, when there is a large volume of material flow between them;
3. **Distance** is constraint used to impose a given relation of distance between two facilities or between a point and a facility. One possible situation occurs when some production units have to operate in a temperature-controlled environment not compatible with others, located in the neighbourhood;
4. **Absolute Position** constraints are used to force facilities to be located, either inside or outside of a given area of the plant the “inside” and “outside”. With these constraints, it is possible to reserve space areas for different purposes like offices or warehouses. These constraints are also used to prevent the location of the facilities in areas that are not inside of the non rectangular plants;
5. **Relative Position** constraints are the ones that make possible handling situations like, for example, "facility A is at right of facility B". There are four possible relative position constrains: ‘at right of’; ‘at left of’; at front of’ and ‘at back of’;
6. **Orientation** constraints deals with situations like the ones that it is necessary to constraint the orientation of a facility or define that several facilities have some kind of relation in terms of its orientation.

A more detailed analysis of these constraint types, especially with the relations that are established between the problem variables, is given in the following subsections. The notation followed to describe the constraints is a simplified form of the previously mentioned, which does not consider the facilities classes.

Preventing the Overlap of the Facilities

As it was mentioned before, the constraint that will be always present is one that inhibits the overlapping of facilities in the plant. Putting this constraint for all the possible pairs of facilities, assures the generation of solutions where facilities do not overlapped, with only a simple labelling procedure applied for all the problem variables. Given two facilities, *i* and *j*, this constraint is given by (23), being the gap value g_{ij} computed by the expression (24).

$$\begin{aligned}&\left(x_i + w_i + g_{ij} \le x_j - w_j\right) \vee \\ &\left(x_j + w_j + g_{ij} \le x_i - w_i\right) \vee \\ &\left(y_i + l_i + g_{ij} \le y_j - l_j\right) \vee \\ &\left(y_j + l_j + g_{ij} \le y_i - l_i\right)\end{aligned} \tag{23}$$

$$g_{ij} = \max\left(g_i, g_j\right) \tag{24}$$

An alternative to formulate this constraint is based on four boolean variables and removes disjunctions that usually gives rise to a bad constraint propagation. This formulation is done with the expressions from (25) to (29).

$$\begin{aligned}&\left(b_{ij}^x = 1 \Leftrightarrow x_i + w_i + g_{ij} \le x_j - w_j\right) \wedge \\ &\left(b_{ij}^x = 0 \Leftrightarrow x_i + w_i + g_{ij} > x_j - w_j\right)\end{aligned} \tag{25}$$

$$\begin{aligned}&\left(b_{ji}^x = 1 \Leftrightarrow x_j + w_j + g_{ji} \le x_i - w_i\right) \wedge \\ &\left(b_{ji}^x = 0 \Leftrightarrow x_j + w_j + g_{ji} > x_i - w_i\right)\end{aligned} \tag{26}$$

$$\begin{aligned}&\left(b_{ij}^y = 1 \Leftrightarrow y_i + l_i + g_{ij} \le y_j - l_j\right) \wedge \\ &\left(b_{ij}^y = 0 \Leftrightarrow y_i + l_i + g_{ij} > y_j - l_j\right)\end{aligned} \tag{27}$$

$$\begin{aligned}&\left(b_{ji}^y = 1 \Leftrightarrow y_j + l_j + g_{ji} \le y_i - c_i\right) \wedge \\ &\left(b_{ji}^y = 0 \Leftrightarrow y_j + l_j + g_{ji} > y_i - l_i\right)\end{aligned} \tag{28}$$

$$1 \le b_{ij}^x + b_{ji}^x + b_{ij}^y + b_{ji}^y \le 2 \tag{29}$$

Although this last formulation seems initially more complex, it can, however, be more efficient in terms of constraint propagation because CLP(*FD*) solvers are incomplete and, therefore, different forms to specify the same constraint can lead to different level of performance when searching for solutions.

Distance

The constraint distance involves the computation of the distance between two facilities or the computation of the distance of a facility to a given point. This kind of constraints creates a new variable with a domain that is a set of possible values for the distance. An important factor that has to be taking into account is how to measure the distance. Here we use the use two alternatives: the rectilinear and the Euclidean metrics.

Before we can formulate the distance constraint we have to formulate a constraint that gives the absolute values of the difference between the domain variables *u* and *v*. This is done with the expressions (30), (31) and (32). The variables b^+ and b^- are two auxiliary boolean variables.

$$|u - v| = \left(b^+ \times u\right) - \left(b^+ \times v\right) + \left(b^- \times v\right) - \left(b^- \times u\right) \tag{30}$$

$$\left(b^+ = 1 \Leftrightarrow u \ge v\right) \wedge \left(b^+ = 0 \Leftrightarrow u < v\right) \tag{31}$$

$$\left(b^- = 1 \Leftrightarrow u < v\right) \wedge \left(b^- = 0 \Leftrightarrow u \ge v\right) \tag{32}$$

It is now possible to formulate the distance constraint. We start by formulating the distance constraint between two facilities, but before we do so, we have to define how to measure the distance. We deal with two situations. In the first one we measure the distance between the centre point of the two facilities and in the second one we measure the distance considering the facilities near edges. Fig 4 illustrates these two situations.

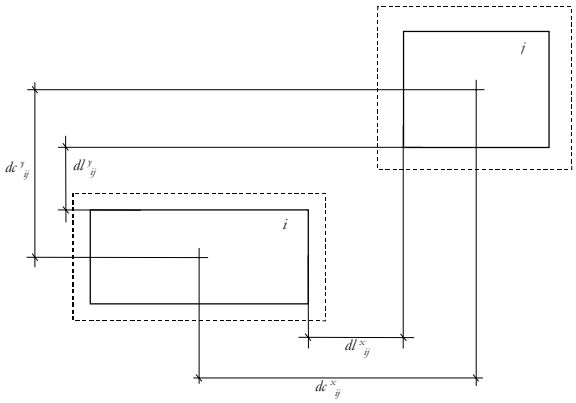


Fig 4: Distance between the centre and considering the edges of two facilities.

Starting with the first situation, the formulation of the distance relation between the centres of two facilities is given by the expressions (33) and (34) in the *x* and in the *y* coordinates respectively. The expression (35) gives the actual distance using a rectilinear metric and (36) gives the euclidean distance.

$$dc_{ij}^{x} = \left| x_i - x_j \right| \quad (33)$$

$$dc_{ij}^{y} = \left| y_i - y_j \right| \quad (34)$$

$$dc_{ij} = dc_{ij}^{x} + dc_{ij}^{y} \quad (35)$$

$$dc_{ij} \times dc_{ij} = dc_{ij}^{x} \times dc_{ij}^{x} + dc_{ij}^{y} \times dc_{ij}^{y} \quad (36)$$

In the second situation, the distance in relation to the near edges of the facilities is slightly more complex to formulate. In this case the distance takes into account the length and the width of the facilities. To better understand how the distance computation is done it is necessary to observe the three possible forms of disposal of two facilities shown in the Fig 5. In the first one the facility *i* is completely above of the facility *j*, and in the second one the facility *i* is completely bellow the facility *j*, and therefore, the distance in *y* is different from zero. In the third situation the distance in *y* is zero because none of the facilities is completely above or completely below of the other. A similar analysis can be made for distances in *x*.

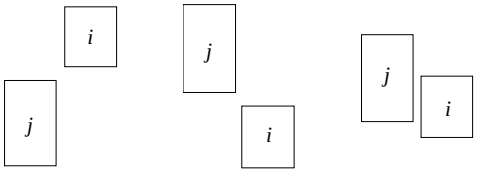


Fig 5: Three cases of relative positions for two facilities in order to compute the distance in relation to the near edges. The total distance is the (a) e (b) sum of the distances in *x* and *y* (c) distance in *x*.

We can say that two facilities are separated in *y* if one is completely above or below of the other. In the same way we can say that two facilities are separated in *x* if one is completely at left or at right of the other.

To establish this relation of distance it is computed, in first place, the distance in *x* and in *y* without taking into account the separation in *x* and in *y* of the two facilities. These distances are given, respectively, by the expressions (37) and (38).

$$de_{ij}^{x} = \left| x_i - x_j \right| - w_i - w_j \quad (37)$$

$$de_{ij}^{y} = \left| y_i - y_j \right| - l_i - l_j \quad (38)$$

Next we compute the separation in x and in y of the two facilities. For this, two boolean variables are required. The value of the first one defines the separation in *x* with the expression (39) and the second defines the separation in *y* with the expression (40).

$$\begin{aligned} &\left(b_{ij}^{x} = 1 \Leftrightarrow \left(x_i + w_i < x_j - w_j\right) \vee \left(x_i - w_i > x_j + w_j\right)\right) \wedge \\ &\left(b_{ij}^{x} = 0 \Leftrightarrow \left(x_i + w_i \geq x_j - w_j\right) \wedge \left(x_i - w_i \leq x_j + w_j\right)\right) \end{aligned} \quad (39)$$

$$\begin{aligned} &\left(b_{ij}^{y} = 1 \Leftrightarrow \left(y_i + l_i < y_j - l_j\right) \vee \left(y_i - l_i > y_j + l_j\right)\right) \wedge \\ &\left(b_{ij}^{y} = 0 \Leftrightarrow \left(y_i + l_i \geq y_j - l_j\right) \wedge \left(y_i - l_i \leq y_j + l_j\right)\right) \end{aligned} \quad (40)$$

Finally we are able to compute the real distance between two facilities. The expression (41) establishes the relation of the distance using a rectangular metric and (42) does the same but using the euclidean metric.

$$de_{ij} = \left(b_{ij}^{x} \times de_{ij}^{x}\right) + \left(b_{ij}^{y} \times de_{ij}^{y}\right) \quad (41)$$

$$de_{ij} \times de_{ij} = \left(b_{ij}^{x} \times de_{ij}^{x} \times de_{ij}^{x}\right) + \left(b_{ij}^{y} \times de_{ij}^{y} \times de_{ij}^{y}\right) \quad (42)$$

Having defined the distance constraints between two facilities it is ease to define the distance constraint between a facility and a point. This is done by assuming that a point is facility with a null width and a null length.

Facilities Neighbourhood

In some situations it is desirable to place two facilities side by side. An example of this arises when there is a large flow of materials between two facilities and therefore if they are neighbours the operation cost is smaller. The use of constraints to express this fact allows a significant reduction in the space of solutions. Also, the placing of two facilities side by side can be a requirement of the problem being solved. It may argue that a distance constraint can do the job, but providing a specific one can give a better performance in constraint propagation. The neighbourhood constraint we are formulating appears under two forms. The first one only imposes that two facilities must be placed side by side. The

second is a more restricted form of the first one and is referred as adjacency constraint.

In relation with the first form of the neighbourhood constraint, the formulation is given with expression (43). This formulation assumes that a non-overlapping constraint is always present.

$$\begin{aligned}&\left(x_i + w_i + g_{ij} \geq x_j - w_j\right) \wedge \\ &\left(x_j + w_j + g_{ij} \geq x_i - w_i\right) \wedge \\ &\left(y_i + l_i + g_{ij} \geq y_j - l_j\right) \wedge \\ &\left(y_j + l_j + g_{ij} \geq y_i - l_i\right)\end{aligned} \tag{43}$$

The adjacency constraint imposes a stronger degree of neighbourhood, which imposes that the two facilities involved are placed in a way that the distance between their geometric centres is minimised. The adjacency constraint can be formulated by the expression (44). Note that *max*(*v*) function gives the biggest value in the domain of *v*.

$$\begin{aligned}&\left(\begin{array}{l} x_i = x_j \wedge c_i = \max\left(w_i\right) \wedge w_j = \max\left(w_j\right) \wedge \\ \left(\begin{array}{l} y_i + l_i + g_{ij} = y_j - l_j \vee \\ y_j + l_j + g_{ij} = y_i - l_i \end{array}\right) \end{array}\right) \vee \\ &\left(\begin{array}{l} y_i = y_j \wedge l_i = \max\left(l_i\right) \wedge l_j = \max\left(l_j\right) \wedge \\ \left(\begin{array}{l} x_i + w_i + g_{ij} = x_j - w_j \vee \\ x_j + w_j + g_{ij} = x_i - w_i \end{array}\right) \end{array}\right)\end{aligned} \tag{44}$$

Once again the use of boolean variables can be used to remove the disjunctions in a similar way done with the non-overlaping constraint.

Facilities Position

There are two types of constraint related with the facilities position that can be defined. The first type, referred as absolute position constraint, imposes that the facilities should be located in specific areas of the plant. The other type, referred as relative position constraint, allows doing the placement of one facility with some relation to another facility.

The absolute position constraints have two forms: one allows the placement of the facilities in some restricted areas of the plant, and the other excludes these areas of the plant for facilities placement. One situation, already referred, using this type of constraints, occurs when the plant shape is not a perfect rectangle. The areas that in the reality do not belong to the plant are excluded using these types of constraints.

The simpler absolute position constraint is the one that imposes that the facility central point should be located at the point $p(x_p, y_p)$. This fact is expressed by (45).

$$x_i = x_p \wedge y_i = y_p \tag{45}$$

The formulation of (45) imposes the placement inside a given area $a(x_a, y_a, w_a, l_a)$, where x_a and y_a represents the geometric centre of the area and w_a e l_a, respectively, are half of its width and half of its length. This more general formulation are given by (46) and (47).

$$\left(x_i - w_i \geq x_a - w_a\right) \wedge \left(x_i + w_i \leq x_a + w_a\right) \tag{46}$$

$$\left(y_i - l_i \geq y_a - l_a\right) \wedge \left(y_i + l_i \leq y_a + l_a\right) \tag{47}$$

The other absolute position constraint is the one that excludes plant areas for the facilities placement. The formulation is done as a logical negation of (46) and (47) that will generate disjunctions. We choose here to present a formulation that removes these disjunctions by using boolean variables. The formulation of this constraint uses the expressions (48) to (52).

$$\begin{aligned}&\left(b_{ia}^x = 1 \Leftrightarrow x_i + w_i < x_a - w_a\right) \wedge \\ &\left(b_{ia}^x = 0 \Leftrightarrow x_i + w_i \geq x_a - w_a\right)\end{aligned} \tag{48}$$

$$\begin{aligned}&\left(b_{ai}^x = 1 \Leftrightarrow x_i - w_i > x_a + w_a\right) \wedge \\ &\left(b_{ai}^x = 0 \Leftrightarrow x_i - w_i \leq x_a + w_a\right)\end{aligned} \tag{49}$$

$$\begin{aligned}&\left(b_{ia}^y = 1 \Leftrightarrow y_i + l_i < y_a - l_a\right) \wedge \\ &\left(b_{ia}^y = 0 \Leftrightarrow y_i + l_i \geq y_a - l_a\right)\end{aligned} \tag{50}$$

$$\begin{aligned}&\left(b_{ai}^y = 1 \Leftrightarrow y_i - l_i \geq y_a + l_a\right) \wedge \\ &\left(b_{ai}^y = 0 \Leftrightarrow y_i - l_i \leq y_a + l_a\right)\end{aligned} \tag{51}$$

$$1 \leq b_{ia}^x + b_{ai}^x + b_{ia}^y + b_{ai}^y \leq 2 \tag{52}$$

In relation with the relative position constraints we find four possibilities: ‘at front of’, ‘at right of’, ‘at back of’ and ‘at left of’. The formulation of these constraints is very simple. However, it is necessary to distinguish partial relative positions from complete relative positions. In the first case only coordinate variables are involved while, in second case, also the facilities width and length variables are involved. The formulation for the first case is done with one of the expressions (53), (54), (55) or (56). The formulation for complete relative position constrains uses one of the expressions (57), (58), (59) or (60).

$$x_i > x_j \quad (53)$$

$$x_i < x_j \quad (54)$$

$$y_i > y_j \quad (55)$$

$$y_i < y_j \quad (56)$$

$$x_i - w_i > x_j + w_j \quad (57)$$

$$x_i + w_i < x_j - w_j \quad (58)$$

$$y_i - l_i > y_j + l_j \quad (59)$$

$$y_i + l_i < y_j - l_j \quad (60)$$

Facilities Orientation

The orientation of the facilities in the plant is controlled with the orientation constrains. Like with position constrains, we have absolute and relative orientation constraints. The first ones usually involve only one facility while the second ones involve at least two facilities. It is defined that a facility is orientated in *x* if the larger edge is parallel to *x*-axis and the formulation correspondent constraint is done with the expression (61). In the same way a facility is orientated in *y* if the larger edge is parallel to *y*-axis and the expression (62) specifies this.

$$\left(b_i^x = 1 \Leftrightarrow c_i > l_i\right) \wedge \left(b_i^x = 0 \Leftrightarrow c_i \leq l_i\right) \quad (61)$$

$$\left(b_i^y = 1 \Leftrightarrow c_i < l_i\right) \wedge \left(b_i^y = 0 \Leftrightarrow c_i \geq l_i\right) \quad (62)$$

The given formulation of the absolute orientation constraints makes ease to formulate the relative orientation constraints. Their formulation uses the boolean variables created by the absolute orientation constraints. So, the formulation of the constraint that imposes the facility *i* to have the same orientation of the facility *j* is given by the expression (63). On the other hand, the formulation to impose a different orientation is given by the expression (64).

$$\left(b_i^x = b_j^x\right) \vee \left(b_i^y = b_j^y\right) \quad (63)$$

$$\left(b_i^x = b_j^y\right) \vee \left(b_i^y = b_j^x\right) \quad (64)$$

4. Problem Solving

Solving a problem, with the CLP(*FD*) paradigm, usually involves at least three steps: the definition of the problem decision variables and their domain; the assertion of the problem constraints; and finally the enumeration of the solution, which instantiates the variables, one by one, with a value from their domain. If the solving task is to find the best solution, or at least a good one, a cost function should also be defined and an optimisation method should be selected.

Two systems were developed which follows these steps as represented in the diagram of the Fig 6. They are the LaRLo /14/, /15/, /18/ and LayGeRL /16/, /17/, /18/ systems.

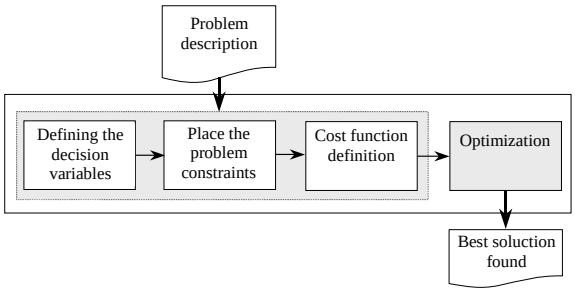


Fig 6: An CLP application typical architecture.

LaRLo system was developed first and uses a Branch & Bound (B&B) algorithm. LayGeRL system is based on Genetic Algorithms (GA) and was developed in order overcome the inability of B&B to find optimal solutions in a practical time period for complex problems.

4.1 LaRLo System

The implementation of the B&B algorithm used in the LaRLo system takes two arguments: a label procedure and the cost function as a Logic Programming (LP) term. The label procedure is used to generate solutions and the cost function is used to place a new constraint when the label procedure finds a solution satisfying all the constraints. This new constraint assures that the next solution being explored is discarded as soon as the cost of the partial solution being explored is already equal or greater than the best valid and complete found solution.

Five label procedures where developed. Fig 7 shows one of the simplest label procedures (*LabelProc*1) written in a logic programming like language. The others four are presented in appendix. It takes a list {Φ} of all problem decision variables grouped by the respective facility and a value λ that specifies the order in which the values in the domain of the variables are instantiated. Each element in {Φ}

represents a facility *k* of class *c* and is in the form of $(i(c,k), r(X_{ck}, Y_{ck}, W_{ck}, L_{ck}, g_{ck}))$.

```
label( {Φ}, λ ) ←
   label_wl( {Φ}, λ ),
   label_xy( {Φ}, λ ).

label_cl( [ ], _ ).
label_cl( [ ( _, r( _, _, W, L, _ ) ) | T ], λ ) ←
   indomain( W, λ ),   indomain ( L, λ ),
   label_cl( T, λ ).

label_xy( [ ], _ ).
label_xy( [ ( _, r( X, Y, _, _, _ ) ) | T ], λ ) ←
   indomain( X, λ ),   indomain( Y, λ ),
   label_xy( T, λ  ).
```

Fig 7: A simple labelling procedure.

The system supports four possible values for the λ parameter (*min*, *middle*, *max* and *partition*).

This approach showed that the exploration of all solution space is most of the times prohibitive, however it is possible to stop the used B&B at the end of a specified time period in order to use the best solution found.

Tab 2: Four different value ordering heuristics.

Domain	1 .. 10
min	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
middle	{6, 5, 7, 4, 8, 3, 9, 2, 10, 1}
max	{10, 9, 8, 7, 6, 5, 4, 3, 2, 1}
partition	{6, 3, 2, 1, 5, 4, 9, 8, 7, 10}

4.2 LayGeRL System

As referred, the LayGeRL system differs from LaRLo in the optimisation technique used. It uses a GA in combination with the CLP paradigm. This approach was inspired in the work done in order to hybridise the B&B algorithm with GA /22/ by following three main principles: use current problem encoding; hybridise if and where possible; and adapt the genetic operators /23/. In this work the GA operators are implemented in CLP as illustrated in Fig 8. The main process is on the GA side. This process can be viewed as the client and the CLP engine can be viewed as the server, which deals with logic and constraint reasoning. The main process needs to start the CLP engine to be able to use its services as shown in Fig 9. When the CLP starts, it begins by creating the problem variables with their finite domain, and then places the constraints according to the problem specifications. This is the start up state (Π) of the CLP engine.

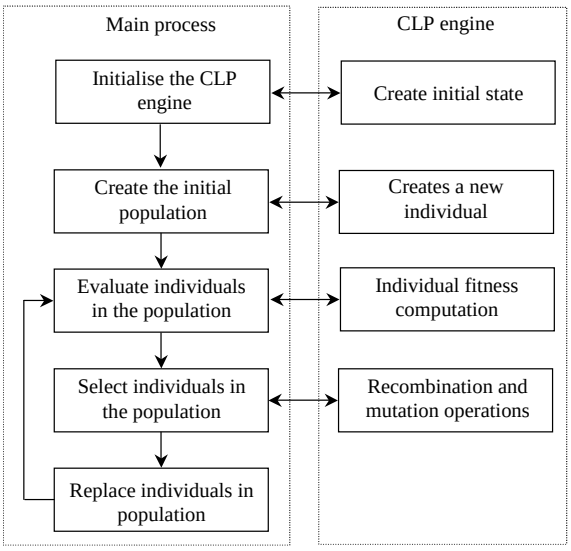


Fig 8: Outline of the CLP and GA combination.

```
procedure MainApp
begin
    <initialisation stuff>
    Π ← clp_startup
    Min ← GA_Optimise(Π, Parameters)
    <exit stuff>
end
```

Fig 9: The main process source code skeleton.

The main process uses the Π returned by the CLP engine to be able to create its initial population, perform the recombination and mutation operations, and finally, evaluate the individuals, which is required to the optimisation task. The individuals produced by the CLP engine represent solutions that must be consistent with the problem constraints placed during the CLP engine start up.

Once the CLP engine is started, the optimisation task begins. This task is a GA like the source code skeleton showed in Fig 10. The operations in italic, with the name starting with *clp_*, are implemented in the CLP paradigm. Although the main process and CLP engine are presented as two separated entities, usually the implementation could be a unique program.

```
procedure GA_Optimise ( Π, Parameters )
begin
    t ← 0
    P0 ← clp_create_initial_population
    clp_evaluate ( Π, Parameters, P0)
    while not Final Condition do
        Pt' ← select_from Pt
        Pt'' ← clp_crossover ( Π, Parameters, Pt')
        Pt''' ← clp_mutate ( Π, Parameters, Pt'')
        clp_evaluate ( Π, Parameters, Pt''')
        Pt+1 ← replace (Pt, Pt''')
        t ← t + 1
    end
end
```

Fig 10: The GA skeleton with operators implemented using the CLP paradigm.

The Genotype Representation

Once the CLP engine executes all the GA operators, the representation of the solutions is done directly using the LP data structure syntax. Each individual in the GA population is only a reference to its respective LP representation. The genotype of each individual is a list of genes, where each one contains information about the respective facility. It is assumed that the genes are always in the same list order.

This data structure with finite domain variables is used as a template to build the individuals during the GA evolution and is created when the CLP engine is started.

Recombination

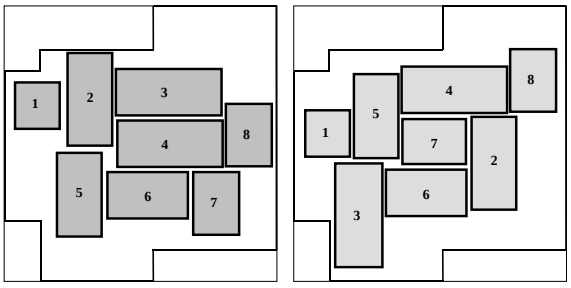
As seen above, the CLP engine executes the recombination of individuals. In a certain way the developed recombination operator performs a slightly form of mutation to ensure that the result of this operator will be consistent with the problem constraints.

The recombination starts by breaking the parent genotype in two random halves. The length of the two halves and the genes in each half are also random. After breaking the parents in two halves, the recombination operation is carried out. As referred above, this operation has to guarantee that the generated offsprings are consistent with the problem constraints. However, it may happen that the recombination operation fails to generate an offspring. This happens when the operator cannot locate the facilities (genes) of the second half in the available space of the manufacturing plant, given the location of the facility of the first half and the problem constraints. A null fitness value is assigned to those failed offsprings and they die before the next generation.

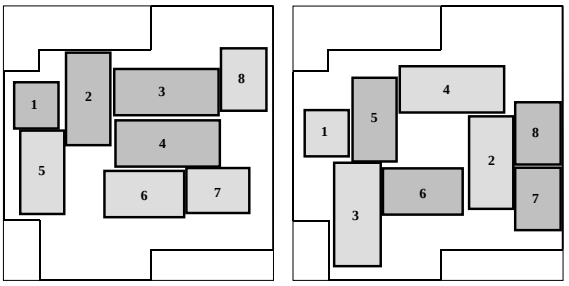
The crossover operation is performed in two stages for each offspring:

1. Locate all the facilities (the correspondent gene) from the larger half in the same location as in a parent. Once the parent is consistent, the partial solution represented by these genes is also consistent;
2. Locate the remaining facilities included in the shorter half from the other parent in the available spaces. It is desirable to place the facilities as close as possible in relation to the locations of the same facilities of the second parent. This is done like a typical CLP labelling method, until a complete solution is found.

The width and length values of all the facilities will remain unchanged. Fig 11 illustrates an example of this recombination operator.



First Parent Second Parent



First Child Second Child

Fig 11: The recombination operation.

Mutation

The effect of the mutation operator is to modify one or more genes of an individual representing a solution. As it was

referred in the previous section the recombination operator has a side effect, which consists in one kind of mutation. This kind of mutation modifies slightly the position of some facilities. However, it is desirable that, from time to time, the orientation or the shape of the objects gets also modified. Among different possible mutation operators, we selected the one that operates as follows:

1. Collect a set of *n* (*n* is a random value) facilities, with *n* less than the cardinality of the genotypes;
2. Modify the width (length) value of the facilities in this set;
3. Place the selected facilities in same position as it was before;
4. If is not possible to place in the same position place in anther available position.

Fig 12 shows an example of two random selected genes for mutation. The shape of the respective facilities is modified by the mutate operator.

Individual Mutated Individual

Fig 12: The result of a mutation operation.

Genetic Operators and the other Constraint Types

The described genetic operators were developed having mainly in mind the non-overlap constraints. When other kinds of constraints are present, like the problem specific constraints placed by the user, the generation of new individuals is more problematic. These other kinds of constraints are always unary constraints (involving only one facility) and binary constraints (two facilities involved). The strategy followed was to design the genetic operators in order to keep the pairs of facilities related by binary constraints in the same half, when breaking a genotype in two halves. Then as a heuristic the placement of facilities is done by first place the facilities participating in more constraints from those that are not already placed in the same location as in their parents.

4.3 Test Problems and Results

The two developed systems were tested with some test problems. Here we present the results obtained with five of the test problems. Their main characteristics are presented in the Tab 3.

Tab 3: Test problems main characteristics.

Problem	pl8	Pl10	pl10c	pl15	pl24
Number of Facilities	8	10	10	15	24
Variable Shape	no	yes	yes	yes	some
Specific Constraints	no	no	yes	no	no
Rectangular Plant Shape	yes	no	no	yes	no

The systems implementation was done using the ECLiPSe system /23/ mainly for the CLP stuff. In the case of the LayGeRL system it was used also the GALib /24/ to write the GA responsible by the optimisation task. The GA implemented is a Steady State GA with overlapping populations. It uses a linear scaling and roulette wheel selection. The termination criterion makes the GA stop when one of two conditions becomes true. One condition is related with the maximum number of generations and the other is true when the standard deviation of the population scores is less than 0.01% of the best individual score in the population.

The first experiments were done with LaRLo system. Each test problem was solved with all combination of labelling procedures with the four different value ordering heuristics. Each experiment ran about one hour. After that time the best solution found was returned. The main goal of these experiments was to try to come across with which combination of labelling procedure and value ordering heuristic tends to explore more promising regions of the search space early and, thus, better solutions. This is important when the B&B algorithm is stopped before the complete search space has been explored.

With these first experiments it was verified that none of the combinations of labelling procedures with the value

ordering heuristics showed to be significantly better than the others. The right combination seems to depend on the problem being solved. Each combination starts with the exploration of the search space at different regions. This makes LayGeRL, with the GA, more adequate as general method to solve this kind of problems since it is capable to explore promising regions without the need to explore the complete search space. The drawback is that there is no guarantee that the best solution is found. On the other hand, the LaRLo system suffers of the same problem, since it stops before the complete exploration of the search space.

The best solutions (low cost) found, by using both systems, are presented in the Tab 4. In parenthesis below the cost value is the processing time, in seconds, used to obtain the respective solutions. Tab 4 shows that LayGeRL system gives always better solutions than LaRLo system, when considering similar processing times.

By analysing the best solutions obtained it was observed that there is a trend to locate facilities close to each other if they have a large material flow between them. This observation suggested that the adjacency constraint should be imposed for some facility pairs in order to improve the performance and the solution quality. Some experiments were made and for most cases it was showed that there was a solution quality improvement. Moreover, the adjacent pairs have to be selected with care since this arbitrary selection can frequently transform the problem in an over-constrained one, which has no solutions. In order to overcome this issue it was developed a systematic method to select the adjacent pairs. This method is based in the computation of the maximum weight matching (MWM) /25/. The concept consists in creating a graph with the production units as nodes. There is an arc connecting two nodes if there is material flow between the respective facility and the weight is the flow volume between them. The maximum weight matching of the created graph is a set of pairs obeying the following conditions:

1. One node participates in only one pair;
2. It is not possible to add a pair without breaking the previous condition;
3. The pairs have a maximum weight sum.

The cost of the solutions obtained by solving the test problems, using both LaRLo and LayGeRL, imposing adjacency constraints between pairs of facilities computed by MWM method is showed in the Tab 5. The labelling procedure used in LaRLo system is similar to *LabelProc*4 presented in appendix, which differs only by first locating the facilities involved in adjacency constraints. The use of adjacency constraints gives better solutions in almost all situations. But, once again, LayGeRL showed to be better to similar processing times.

5. Conclusions

In this paper we proposed a method to address the facilities layout design problem (FDLP) through the technology of constraint logic programming (CLP). Two prototype systems were developed: LaRLo and LayGeRL. They differ only in the technique used in the optimisation task. CLP is a new technology, with wide potential, based on the logic programming and computational processes that appeal to the imposed constraints on the problem variables. In the LayGeRL system we also look for an hybrid approach using CLP with genetic algorithms (GA). The developed system looked for the advantages, on one hand, of a process imminently abstract and declarative for the specification of problems and, on the other hand, the potentialities that the evolutionary computation offers in the attainment of solutions, mainly when there is no specific methods to solve the problem in a proper way. As it was showed in this paper, the generation of industrial plant layout is indeed a complex optimisation problem, where it has to focus to a set of several constraints imposed on the problem variables.

Being the FDLP a complex problem, in particular the model presented, it was also showed that the exploration of all the search space is not practical for real problem and, thus, the branch and bound algorithm is not the most adequate optimisation technique. The use GA showed to be a technique offering a good compromise between the amount of the search space that is explored, the quality of solutions and the performance.

Another important aspect that can be retained from this work is that the combination of CLP and GA is not limited to be applied to the FLDP. It can be applied to solve other problems. To apply this framework it is only necessary to define the genetic operators according to the problem structure. This can be advantageous once the developer can use the problem structure to get specific and well adapted genetic operators. On the other hand, this approach has the disadvantage of reducing the robustness of the GA, because the genetic operators are more dependent of the problem structure.

The developed system suffers from some limitations that we hope in the future deal with. In general they can be viewed in two research domains: the model of the problem in order to deal with the new trend of the manufacturing systems and the CLP technology used to solve the problems. With the first research domain we can point, as example, the facilities that cannot be always modelled as rectangular shapes, the manufacturing plant has a third dimension (several floors) and the constraint are not always mandatory (they can have levels of priorities). The limitations of the technology are almost all related with the performance. In order improve this several paths can be followed. For example, the development of global constraints in order to get better quality in constraint propagation, more efficient and intelligent label procedures – LaRLo – and genetic operators – LayGeRL. Distributed GA and the cooperation with both systems are also issues for further work.

Tab 4: The computational results of the two systems in presence of the test problems without adjacency constraints (PS - population size, RR – replacement rate, RP – recombination probability, MP – mutation probability).

Problem		**pl8**	**pl10**	**pl10c**	**pl15**	**pl24**
LaRLo	Cost	31377 (3454)	25836 (3444)	25926 (2126)	29286 (3427)	109372 (2727)
	λ	Min	Middle	Min	Min	Middle
	Label Procedure	*LabelProc*5	*LabelProc*5	*LabelProc*5	*LabelProc*5	*LabelProc*1
LayGeRL	Cost	24239 (85)	21653 (611)	23161 (644)	25270 (3797)	96232 (1352)
	PS/RR/RP/MP	100/0,4/1,0/0,2	100/0,4/1,0/0,2	100/0,5/1,0/0,15	100/0,4/1,0/0,25	80/0,1/0,8/0,05

Tab 5: The computational results of the two systems in presence of the test problems with adjacency constraints (PS - population size, RR – replacement rate, RP – recombination probability, MP – mutation probability).

Problem		**pl8**	**pl10**	**pl10c**	**pl15**	**pl24**
LaRLo	Cost	22784 (825)	23422 (905)	22985 (1056)	29361 (499)	114401 (1688)
	λ	Partition	Partition	Min	Middle	Min
LayGeRL	Cost	22559 (45)	18734 (1070)	18925 (2576)	25745 (9033)	94911 (6465)
	PS/RR/RP/MP	100/0,4/1,0/0,15	100/0,5/0,9/0,15	100/0,5/0,9/0,15	100/0,5/0,9/0,15	60/0,1/0,8/0,05

REFERENCES

1. Armour, G.C. and Buffa, E. S. 1963. "A heuristic algorithm and simulation approach to relative location of facilities". *Management Science*, 9 pp 294-309.
2. Vollman, T. E. and Buffa, E. S.: 1966. "Facilities layout problem in perspective". *Management Science*, 12 pp 450-468.
3. Riggs, J. L. 1987. *Production Systems: planning, analysis*, and *control*. Fourth Edition, John Wiley & Sons, Inc, ISBN 0-471-85888-9.
4. Heragu, S. S. and Kusiak, A., 1987. "The Facility Layout Problem", *European Journal of Operational Research*, 53, pp 1-13.
5. Kusiak, A. 1990. Intelligent *Manufacturing Systems*, Prentice Hall. Englewood Cliffs, NJ, ISBN 0-13-468364-1.

6. Meller, R. D. and Gau, K.-Y. 1996. "The Facility Layout Problem: Recent Trends and Perspectives", *Journal of Manufacturing Systems*, 15(5), pp 351-366.

7. Heragu, S. 1997. *Facilities Design*. PWS Publishing Company, ISBN 0-534-95183-X.

8. Dimopoulos, Christos and Zalzala, Ali M. S., 2000. "Recent Developments in Evolutionary Computation for Manufacturing Optimization: Problems, Solutions and Comparisons". IEEE Transations on Evolutionary Computation, 4(2), pp 93-113.

9. Jaffar, J., Lassez, Jean-Louis, A. 1987. "Constraint logic programming". *In Proceedings of the 14th ACM Symposium on Principles of Programming Languages*, Munich, Germany, ACM, January, pp 111-119.

10. Kumar, V. 1992. *Algorithms for Constraint Satisfaction Problems: A Survey*. Department of Computer Sciences, University of Minnesota.

11. Frühwirth, T., Herold, A., Küchenhoff, V., Provost, T., Lim, P., Monfroy E. and Wallace, M. 1993. *Constraint Logic Programming - An Informal Introduction*. European Computer-Industry Research Centre.

12. Koopmans, T. C., and Beckman, M. 1957. "Assignment Problems and the Location of Economic Activities". *Econometrica*, 25, pp 53-76.

13. Montreuil, B. H., Venkatadri, U., e Ratliff, H. D. 1993. "Generating a layout from a design skeleton". *IIE Transactions*, 25(1), pp 3-15.

14. Tavares, J., Ramos, C. and Neves, J. 1999, "Constraint Programming Approach to Solve Facility Layout Design Problems", *ISATP'99 – 1999 IEEE International Symposium on assembly and Task Planning*, Porto, Portugal, pp 368-373.

15. Tavares, J., Ramos, C. and Neves, J. 1999, "A Model to Solve the Facility Layout Problem Using Constraint Logic Programming". *IMS'99 – Second International Intelligent Manufacturing Systems 1999*, Leuven, Belgium, pp 429-438.

16. Tavares, J., Ramos, C. and Neves, J. 2000, "Using Genetic Algorithms in a Constraint Programming Framework to Optimise Facility Layout Design Problems". *PACLP'2000 – The Practical Application of Constraint Technologies and Logic Programming*, Manchester, UK, pp 41-58.

17. Tavares, J. Ramos, C. and Neves, J. 2000, "Addressing the Layout Design Problem Through Genetic Algorithms and Constraint Logic Programming". *ASC'2000 – Third IASTED International Conference of Artificial Intelligence and Soft Computing*, Banff, Alberta, Canada, pp 65-71.

18. Tavares, José. 2002. *Generation of Industrial Systems Configurations by Using the Constraint Technology and Evolutionary Computation*. PhD Thesis, Dept. of Informatics, School of Engineering, University of Minho, Portugal (in Portuguese).

19. Davis, L. 1991. *A Genetic Algorithms Tutorial. In Handbook of Genetic Algorithms*, L. Davis (ed), New York, USA: Van Nostrand Reinhold. pp. 1-101.

20. Michalewicz, Z. 1996. *Genetic algorithms + Data Structures = Evolution Programs*, Springer-Verlag, New York, third edition.

21. Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. Univ. of Michigan Press, Ann Arbor, MI.

22. Cotta, C., Aldana, J.F., Nebro, A.J., and Troya, J.M. 1995. Hybridizing Genetic Algorithms with Branch and Bound Techniques for *the* Resolution of the TSP. *Artificial Neural Nets and Genetic Algorithms*, D.W. Pearson, N.C. Steele, R.F. Albrecht (eds.), Springer Verlag Wien - New York, pp 277-280, ISBN 3-211-82692-0.

23. Schimpf, J., Brisset, P., Sakkout, H., Frühwirth, T., Gervet, C., Meier, M., Novello, S., Provost, T., Shen, K., and Wallace, M. 1999. *ECLiPSe 4.2 User Manual*. International Computers Limited and Imperial College London. HTTP://www.icparc.ic.ac.uk/eclipse/

24. Wall, Matthew. 1996. *Galib - A C++ Genetic Algorithms Library Users* Manual. Mechanical Engeneering Department, Massachusetts Institute of Technology.

25. Lengauer, T. 1990. *Combinatorial Algorithms for Integrated Circuit Layout*. Chichester, UK: John Wiley & Sons Ltd.

APPENDIX

```
label( [ ], _ ).
label( [ ( _, r( X, Y, W, L, _ ) ) | T ], λ ) ←
   indomain( W, λ ),   indomain( L, λ ),
   indomain( X, λ ),   indomain( Y, λ ),
   label( T, λ ).
```

Fig 13: Pseudo-Prolog code for *LabelProc*2.

```
label( {Φ}, λ ) ←
   label_wl( {Φ}, λ ),   label_xy( {Φ}, λ ).

label_wl( [ ], _ ).
label_wl( [ ( _, r( _, _, W, L, _ ) ) | T ], λ ) ←
   indomain( W, λ ),   indomain( L, λ ),
   label_wl( T, λ ).

label_xy( [ ], _ ).
label_xy( {Φ}, λ ) ←
   remove_pf( ( _, r( X, Y, _, _, _ ) ), {Φ}, T ),
   indomain( X, λ ),   indomain( Y, λ ),
   label_xy( T, λ ).
```

Fig 14: Pseudo-Prolog code for *LabelProc*3.

```
label( {Φ}, λ ) ←
   collect_all_pairs( Pairs ),
   sort_pairs( Pairs, Sort_Pairs ),
   label( {Φ}, Sort_Pairs, λ ).

label( _, [ ], _ ).
label( {Φ}, [ φ( I_i , I_j , _)|Pairs ],λ ) ←
   Φ( I_i , r( X_i, Y_i, W_i, L_i, _ ) ),
   Φ( I_j , r( X_j, Y_j, W_j, L_j, _ ) ),
   domain_size( X_i, TX_i ),
   domain_size( Y_i, TY_i ),
   domain_size( X_j, TX_j ),
   domain_size( Y_j, TY_j ),
   (
      TX_i × TY_i ≤ TX_j × TY_j ,
      !,
      label( W_i, L_i, X_i, Y_i, λ ),
      label( W_j, L_j, X_j, Y_j, λ ),
   ;
      label( W_i, L_i, X_i, Y_i, λ ),
      label( W_j, L_j, X_j, Y_j, λ ),
   ),
   label( T, Pairs, λ ).

label( W, L, X, Y, λ ) ←
   indomain( W, λ ),
   indomain( L, λ ),
   indomain( X, λ ),
   indomain( Y, λ ).

collect_all_pairs( Pairs ) ←
   findall( φ( I_i , I_j , F_ij), φ( I_i , I_j , F_ij), Pairs ).
```

Fig 15: Pseudo-Prolog code for *LabelProc*4.

```
label( [ ], _ ).
label( {Φ}, λ ) ←
   delete( (_, r( X, Y, C, L, _ ) ), {Φ}, T ),
   label( C, L, X, Y, λ ),
   label( T, λ ).

label( C, L, X, Y, λ ) ←
   indomain( C, λ ),    indomain( L, λ ),
   indomain( X, λ ),    indomain( Y, λ ),    !.
```

Fig 16: Pseudo-Prolog code for *LabelProc*5.