Encoding Structures and Operators used in Facility Layout Problems with Genetic Algorithms

García-Hernández, L.[a], Araúzo-Azofra, A.[a], Pierreval, H.[b] and Salas-Morera, L.[a]

(a) Area of Project Engineering
University of Cordoba (Spain)
ir1gahel@uco.es, arauzo@uco.es, lsalas@uco.es

(b) LIMOS UMR CNRS 6158 IFMA (France)
Henri.Pierreval@ifma.fr

***Abstract*—The allocation of facilities in a plant layout is a complex problem. For solving it, many authors have used Genetic Algorithms (GAs) with the objective of reaching an efficient plant layout design. To represent the plant layout design as a data structure, GAs require a defined encoding scheme. Such a structure defines the types of solutions that can be obtained, and influences the GA´s ability to find good solutions. There are a few surveys on facility layout problems, but they have not addressed evolutionary issues in depth. This work presents a review that focuses on encoding schemes and related operators used in GAs, and suggests a method of classifying the different encoding structures described in the bibliography. We also studied their main characteristics and objectives; and successfully identified the crossover and mutation operators that could be utilized depending on the type of encoding scheme.**

Keywords—Facility layout problems; Encoding schemes; Genetic Algorithms.

I. INTRODUCTION

Facility Layout Design determines the placement of facilities (sometimes called departments) in a manufacturing plant with the aim of achieving the most effective arrangement in accordance with some criteria or objectives laid down, while also admitting some constraints. Among others, these objectives could be to minimize the material handling cost, to maximize the closeness of relationships between each pair of facilities, or to satisfy a desired aspect ratio. Plant Layout Design is crucial for attaining production efficiency [16] because it directly influences manufacturing costs, lead times, work in process and productivity. Well laid out facilities contribute to the overall efficiency of operations and could reduce between 20% and 50% of the total operating costs [32].

Many techniques have been applied to deal with Plant Layout Design. One of those most widely used is Genetic Algorithms (GAs). An essential step in building a GA is to decide on the genetic representation of an individual (genotype), which must be concordant with a candidate solution of the problem (phenotype) on applying the decoding procedure. An important and difficult part of designing a good GA [9] is choosing the appropriate encoding scheme, as the choice also determines the operators such as crossover or mutation that could be applied.

This paper presents an overview of encoding schemes and evolutionary operators used in GAs found effective in solving Facility Layout Problems. Indeed, they are important for a GA´s ability to evolve good solutions. First, in Section 2, we define layout problems. Section 3 describes the manner in which the facilities could be placed in the layout. In Section 4, the emphasis is on creating facility layout solutions. Section 5 discusses ways to encode solutions. Section 6 examines the evolutionary operators associated with each encoding scheme.

II. FACILITY LAYOUT PROBLEMS

Plant layout problems are of several kinds [7], and are solved by employing different techniques [17, 27, 24]. To design a plant layout, several characteristics that define the different problems needing solutions are to be considered. Some important features to take into account are:

- Facility shape. It can be regular(all the facilities have the same shape, e.g. a rectangle) [1, 2, 4, 5, 6, 8, 10, 11, 12, 14, 18, 19, 20, 21, 22, 25, 28, 29, 30, 31, 34] or irregular (e.g. a polygon) [3, 13, 15, 26, 33].
- Facility dimensions. These can be equal [2, 5, 6, 10, 21, 25, 26, 29, 31], when all the facilities have the same dimensions and size, or unequal [1, 3, 4, 8, 11, 12, 13, 14, 15, 18, 19, 20, 22, 26, 28, 30, 33, 34], when at least one of them is different.
- Number of floors. Most authors have considered only a single floor. But some of them [20, 22] have considered several floors (multi-floor facility layout problem). Which complicate the plant design because they

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.206

introduce more constraints (e.g. Vertical distance) and/or devices (stairs, elevators).

- Planning horizon. A problem is defined as static when the environment is stable and there is no change in production requirements. It is dynamic [2, 6, 8] when the manufacturing plant is designed to enable it adapt the plant to a changing environment.
- Aspect ratio. This value is applied to penalize impracticable solutions (such as when in the solution facilities appears with high length value and very small width value).
- Objective function. Objective concerns are the factors to be optimized. The objective function allows to evaluate each solution to determine its performance.
- Specific features. It is possible to conclude passages or aisles [13, 19, 20, 34], inner walls [19, 20], elevators [22] and stairs, among others.

III. Layout Representations

In the related literature, several types of representations are used to define the manner of placement of the facilities on the surface. There are, in the continuous type, the Bay [1, 4, 11, 13, 18, 19, 20, 23, 30] and the Slicing Tree [14, 22, 28, 34] methods. In the discrete case, the representation used is the Grid [2, 3, 5, 6, 10, 15, 21, 25, 26, 29, 31, 33].

Bays. This approach divides the plant into a number of bay blocks, which is either fixed or variable. It is also possible that the width of all of the bays that make up the layout, are either fixed or variable. Figure 1 is an example taken from [13].

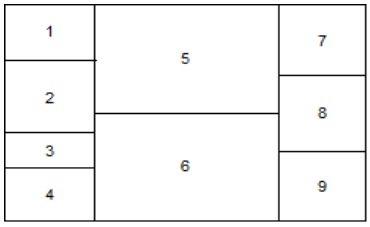

Figure 1. Bay Structure.

Tree Structure. In this type, a slicing tree structure represents the layout: each leaf node represents a facility, and each internal node the slicing operator that cuts the layout into portions or allocations. These operators could be vertical or horizontal cuts, or they could be more detailed such as, below, up, right or left cuts. In Figure 2, we can see the slicing tree and its representation (taken from [28]).

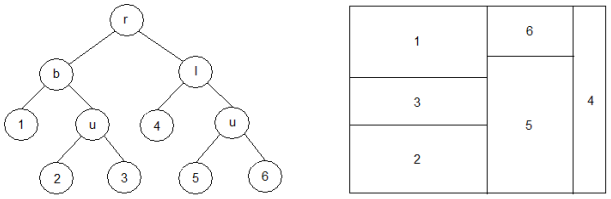

Figure 2. A slicing tree and the slicing structure, respectively.

Grid. This approach divides the plant into squares of the same area and dimensions. If the facilities are of equal dimensions and regular shape, we have only a simple problem of allocating n facilities into m positions. However, if the dimensions are unequal and/or the shape is irregular, it becomes necessary to adopt another structure (e.g. the Space Filling Curve (SFC)), that enables identification of each square within a determined facility (see Figure 3 taken from [3]).

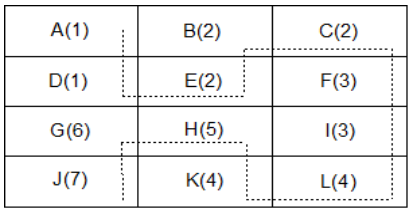

Figure 3. An example of a 12-square layout with SFC.

IV. Creating Solutions

Usually the data structure that represents a facility layout is complex. To improve encoding schemes, in this section, we identify the component functions that lead to a complete structured plant design. Each of these functions could be encoded in different ways that are described in the next section. The identified elemental functions are:

Place. This function places a facility in a location determined by its coordinates. From Figure 4 we can see that the encoding schemes that implement this are the float permutations [8] and float strings without restrictions [19, 20]. In the former, the float strings are permutations of the facility center coordinates. The latter encoding divides the distance proportionately between the origin and the center of aisles.

Sort. This function arranges in order the facilities in the layout of a plant, and determines their sequence (e.g. we have a different sequence of facilities if we read the facility from top to bottom and from left to right than if we read it in the reverse order). From Figure 4, we find that to provide this function, the permutations are normally used. Logically, when integer permutations with integers as operators are intercalated [22], the sequence of the facilities in the layout also appears. In [28] is described another way to sort the order of the facilities. They used, a comprised value between '0' and '1' randomly assigned to each facility for determining the facility sequence by sorting

values. The last encoding, that allows sorting of the elements in the plant is the float string with the restriction that the aggregate of all the string elements is less or equal to the total area of the plant [19, 20].

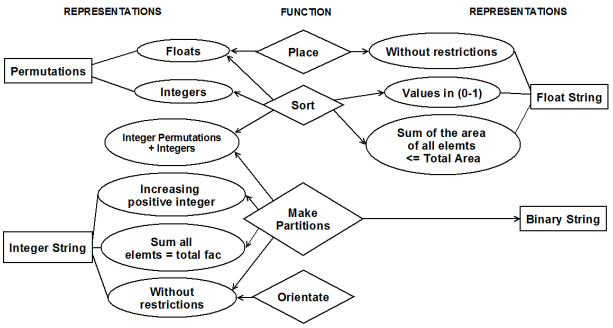

Figure 4. Relations between the functions and their associated encoding schemes.

Partitioning. This function divides the plant into portions (e.g. when the layout is divided into bays). There are several options for encoding this function (see Figure 4). The first of these is the combination of integer permutations and integers. Matsuzaki et al. [22] use this method in the slicing tree, where the integer permutations indicate the facility sequence, and the other integers are operators that enable division of the plant into sections. The second option is a string of increasing positive integers [30], which indicates the locations of the breakpoints. The third way of grouping is through the integer string. In this method, the elements could be inserted without restrictions, being operators that represent the plant divisions [14, 28, 34], or they could show the way to group the facilities through the sweeping direction and the sweeping band [33, 15]. Another grouping option is the integer string, where the element adds the total number of facilities that exists in the layout [1, 11, 4, 20]. This string indicates the number of facilities in each bay block, such that the string has the same elements as the bays in the plant. The last option for grouping is binary encoding [13, 33, 15, 8]. When the value '0' appears in the binary string, it indicates that the equivalent facility is in the same group or block; when the value '1' appears in the string, it is the indication to begin with another block.

Orientation. Orientation is the last function seen in Figure 4. This function allows rotation of a facility over its central axis with respect to the point of origin. For setting the orientation of the facility, a integer string is used [34], in which each element is an operator that provides this facility orientation.

V. ENCODING

Having analyzed the studies that have investigated this aspect, we can classify the encoding schemes into several types:

Integer/Real permutations: The objective of this encoding is to determine the facility sequence that comprises a plant. Generally, this encoding is a string of n sizes, where n is the number of facilities in the layout. Logically, the string can not have repeated elements, because the same facility can not be placed in two locations of the layout. Many authors have employed integer strings to establish the facility sequence [1, 2, 3, 4, 5, 10, 11, 12, 14, 15, 19, 20, 21, 25, 26, 29, 30, 31, 33, 34]. However, only a few have used real strings [19, 20]. In the case of dynamic layout, some authors have represented the corresponding layout for each period with permutations of integers [2, 6] or floats [8]. In the last case, the author uses the real string to establish simultaneously the facility sequence and position the facility centers.

Integer permutations and integers: This encoding was proposed by Matsuzaki et al. [22] to determine the facility sequence and to simultaneously group the facilities in the layout. For the first method, this method uses integer elements that can not be repeated. To represent the cut operators, he uses 4 characters (that can be translated into integers) that can be repeated. Both types of elements are combined in the string. The size of the string is the sum of the number of facilities and the numbers of cut operators (which are equal to the number of facilities minus one).

Integer string: this type of encoding scheme could be divided into three types.

- Increasing positive integer. The integer string is created because each element is larger than the next. Tate and Smith [30] use this encoding to group into bays the elements of the layout, which consists of an integer string where each element represents the last facility of the bay. The total number of elements of the string plus one, is equal to the total number of the bays in the plant.
- Sum of all elements equal to total number of facilities. The integer string is created because each element is greater than '1', and the sum of all elements is equal to the number of facilities that make up the layout. Some authors [1, 4],[11], and [20] have used this encoding to group the facilities into bays, and hence propose an integer string where each integer represents the number of elements in a bay.
- Without restrictions. This encoding is used to group facilities and to determine the orientation of the facility. Honiden [14] employed an integer string to show the grouping order of facilities. Tam [28] used a string composed of characters (that can be translated to integers), each of them of a value of four possible operators (bottom, upper, right, and left), which determine the cuts of the slicing tree. Wang et al. [33] and Hu et al. [15] utilize this coding

scheme to show the sweeping band, that determines the method of grouping the layout. Wu and Appelton [34] uses this encoding for two functions: on the one hand, he employs a string of integers that represents the cutting levels and allows grouping of the facilities in the plant. On the other hand, he uses another integer string to indicate the orientation of the facility. Each element of this string could associate one of four possible values (0º, 90º, 180º and 270º).

Float: this encoding scheme could be divided into three cases, too:

- Without restrictions. The real or float string is used to place the element position in the layout. Lee et al. [19, 20] in order to allocate the vertical and horizontal passages in the plant.
- Values in (0,1). The real or float string is created considering that the value each element has is included in the range (0-1). Norman and Smith [23] use this encoding to arrange the sequence of the facilities of the plant by assigning a random value between '0' and '1' to each facility and then, arranging the string from the smaller to the higher value.
- Sum of all elements are <= total area. The float string is created considering that each element value is included between lower and higher bounds. Moreover, it is necessary that the sum of all elements be lower or equal than the total area of the distribution. In this case, the string is composed of float elements (that are organized as the string of facility sequence) which offer the area information of each facility in the layout.

Binary string: This type of encoding groups the facilities in the layout to enable determination of the orientation of the facility. Gomez et al. [13] employed a binary string of elements to divide the plant into bays. When the value '1' appears in the string, the facility is the last among the bays, in the other case, the value '0' appears in the string. Moreover, Dunker et al. [8] used a binary string to establish the facility orientation in the dynamic layout. If the value is '0', the orientation is vertical, or else, it is horizontal.

VI. Crossover and Mutation operators

The operators analyzed are: Crossover (allows to create children from two or more parents), and Mutation (allows to obtain a new offspring modifying the parent).

Most of the operators analyzed are well known and are illustrated in [9]. The crossover operators studied are Uniform, PMX, OX, CX, N-point, and the selection of the best parent (is taken for the child created).

The studied mutation operators are: PM, AM, SM, Inverse, PM if improved (the mutation is done if the new individual is better), Insert/Delete or Increase/Decrease a gen, Divide or Join genes.

TABLE I. ANALYSIS OF CROSSOVER METHODS.

Representation		Crossover					
Encoding Scheme		Uniform	PMX	OX	CX	N point	Select the best
Permut.	integers	√	√	√	√	√	√
	floats	O	√	O	O	√	O
Int. Permut. + Int.		√	O	O	O	O	O
Int. String	Increasing positive	X	X	X	X	O	O
	Sum all = total fac.	X	√	O	O	X	O
	Without restrictions	O	O	O	O	√	O
Float String	Without restrictions	O	O	O	O	√	O
	(0-1) val.	X	X	X	X	O	O
	Elem. area sum <= total area.	X	√	O	X	X	O
Binary String		O	√	O	X	O	O

We can see these operators, Crossover in Table 1 and Mutation in Table 2, and those studied in the revised works, are marked '√'. The methods that could not be applied by the encoding nature are marked 'X', and finally, we marked 'O' those that can be applied but are not used in reviewed literature.

TABLE II. ANALYSIS OF THE MUTATION METHODS.

Representation		Mutation						
Encoding Scheme		PM	AM	SM	Inv	PM if impr	Ins/ Del or Inc/ Dec a gen	Div/ Join
Permut	Int.	√	√	√	√	√	X	X
	floats	√	O	O	O	O	X	X
Int. Permut. + Int.		O	O	O	√	O	O	√
Int. String	Incr. Positive	X	X	X	X	X	O	X
	Elem. sum = nº fac.	√	O	O	O	O	√	O
	Without restrict.	√	O	O	O	O	X	X
Float String	Without restrict.	O	O	O	O	O	X	X
	(0-1) Values	X	X	X	X	X	X	X
	Elem. area sum <= total area.	X	O	O	O	X	X	X
Binary String		√	O	O	O	O	X	X

VII. Conclusions

In this paper, we have presented a survey that focuses on encoding schemes and the evolutionary operators used by GAs applied to Facility Layout Problems. Other surveys have examined Facility Layout, but have not studied evolutionary techniques in depth. Although, this overview can not be exhaustive, the analysis carried out enables us identify: (1) the manner of placement of facilities on the surface, (2) the component functions that could be used to create the facility layout solutions, and (3) techniques to encode them. Combining the identified component functions could create new unexplored encoding schemes.

In this manner, many different ways of encoding the facility layout solutions are available. Logically, crossover and mutation operators also depend on the encoding scheme selected. Moreover, we have identified the evolutionary operators that could be applied to each encoding scheme. Some of them have not been tested yet. These encoding schemes and their operators will determine the ability of the GA to obtain good solutions.

The classifications and analyses described in this work, could prove useful for future studies in facility layout problems. In this context, the next step of research could be to evaluate new encoding schemes and untested evolutionary operators. This would enable achieve the aim of improving results and recommending the best among them.

REFERENCES

[1] G. Aiello, M. Enea, and G. Galante, "A multi-objective approach to facility layout problem by genetic search algorithm and Electre method", Robotics and Computer-Integrated Manufacturing, Vol. 22, pp.447–455, 2006.

[2] J. Balakrishnan, C.H Cheng, D.J. Conway and C.M. Lau, "A hybrid Genetic Algorithm for the Dynamic Plant Layout Problem", International Journal of Production Economics, Vol 86, pp.107-120, 2003.

[3] J. Balakrishnan, C.H. Cheng and ,K.F. Wong, "FACOPT: A User Friendly FACility Layout OPTimization System", Computers & Operations Research, Vol 30, pp.1625-1641, 2003.

[4] J. Chae and B.A. Peters, "Layout Design of Multi-Bay Facilities with Limited Bay Flexibility", Journal of Manufacturing, Vol 25, 2006.

[5] C. Chan and H. Tansri, "A Study of Genetic Crossover Operations on the Facilities Layout Problem", Computers & Industrial Engineering, Vol 26, pp.537-550, 1994.

[6] D. Conway and M. Ventakaramanan, "Genetic Search and the Dynamic Facility Layout Problem", Computers & Operations Research, Vol 21, pp.995-960, 1994.

[7] A. Drira, H. Pierreval and S. Hajri-Gabouj, "Facility Layout Problems: A Survey". Annual Reviews in Control, Vol 31, pp.255-267, 2007.

[8] T. Dunker, G. Radons and E. Westkämper, "Combining Evolutionary Computation and Dynamic Programming for Solving a Dynamic Facility Layout Problem". European Journal of Operational Research, Vol 165, pp.55-69, 2005.

[9] A.E. Eiben and J.E. Smith, Introduction to Evolutionary Computing. Springer, 2007.

[10] M. El-Baz, "A Genetic Algorithm for Facility Layout Problems of Different Manufacturing Environments", Computers & Industrial Engineering, Vol 47, pp.233-246, 2004.

[11] M. Enea, G. Galante, and E. Panascia, "The Facility Layout Problem Approached using a Fuzzy Model and a Genetic Search", Journal of Intelligence and Manufacturing, Vol 16, pp.303-315, 2005.

[12] M. Ficko, M. Brezocnik and J. Balic, "Designing the Layout of Single- and Multiple-Rows Flexible Manufacturing System by Genetic Algorithms", Journal of Materials Processing Technology, Vol 157-158, pp.150-158, 2004.

[13] A. Gómez, Q.I. Fernández, D. De la Fuente García and P.J. García, "Using Genetic Algorithms to Resolve Layout Problems in Facilities where there are Aisles", International Journal of Production Economics, Vol 84, pp.271-282, 2003.

[14] T. Honiden, "Tree Structure Modeling and Genetic Algorithm-based Approach to Unequal-area Facility Layout Problem", Industrial Engineering & Management Systems, Vol 3, pp. 123-128, 2004.

[15] M.H. Hu, and M.J. Wang, "Using Genetic Algorithms on Facility Layout Problems". International Journal of Advanced Manufacturing Technology, Vol 23, pp.301-310, 2004.

[16] P. Kouvelis, A.A. Kurawarwala and G.J. Gutiérrez, "Algorithms for Robust Single and Multiple Period Layout Planning for Manufacturing Systems", European Journal of Production Research, Vol 63, pp.287-303, 1992.

[17] S. Kuturel-Konak, "Approaches to Incertainties in Facilities Layout Problems: Perspectives at the Beginning of the 21st Century", Journal of Intelligence and Manufacturing, Vol 18, pp.273-284, 2007.

[18] Y.H Lee. and M.H. Lee, "A Shape-based Block Layout Approach to Facility Layout Problems Using Hybrid Genetic Algorithm". Computers & Industrial Engineering, Vol 42, pp.237-248, 2002.

[19] K.Y. Lee, S.N. Han and I.R. Myung, "An Improved Genetic Algorithm for Facility Layout Problems Having Inner Structure Walls and Passages", Computers & Operations Research, Vol 30, pp.117-138, 2003.

[20] K.Y. Lee, M. Roh and H. Jeong, "An Improved Genetic Algorithm for Multi-Floor Facility Layout Problems Having Inner Structure Walls and Passages", Computers & Operations Research, Vol 32, pp.879-899, 2005.

[21] K.L. Mak, Y.S. Wong and F.T.S. Chan, "A Genetic Algorithm for Facility Layout Problem", Computers Integrated Manufacturing System, Vol 11, pp.113–127, 1998.

[22] K. Matsuzaki, T. Irohara, and K. Yoshimoto, "Heuristic Algorithm to Solve the Multi-Floor Layout Problem with Consideration of Elevator Utilization". Computers & Industrial Engineering, Vol 36, pp.487-502, 1999.

[23] B.A. Norman A.E. and Smith, "A Continuos Approach to Considering Uncertainty in Facility Design", Computers & Operation Research, Vol 33, pp.1760-1775, 2006.

[24] H. Pierreval, C. Caux, J.L. Paris, and F. Viguier, "Evolutionary approaches to the design and organization of manufacturing systems", Computers & Industrial Engineering, Vol 44, pp.39-364, 2003.

[25] A.S. Ramkumar, S.G. Ponnambalam, N. Jawahar, and R.K. Suresh, "Iterated Fast Local Search Algorithm for Solving Quadratic Assignment Problems", Robotics and Computer-Integrated Manufacturing, Vol. 24, pp.392–401, 2008.

[26] J. Singh, B.T. Foster and S.S. Heragu, "A Genetic Algorithm for the Unequal Area Facility Layout Problem", Computers & Operations Research, Vol 25, pp.583–594, 1998.

[27] S.P. Singh and R.R.K. Sharma, "A Review of Different Approaches to the Facility Layout Problems", International Journal of Advanced Manufacturing Technology , Vol 30, 2006 , pp.425–433.

[28] K.Y. Tam, "Genetic Algorithms, Function Optimization, and Facility Layout Design". European Journal of Operational Research, Vol 63, pp.322-346, 1992.

[29] D.M. Tate and A.E. Smith, "A Genetic Approach to the Quadratic assignment problem", Computers & Operations Research, Vol 22, pp.73–83, 1995.

[30] D.M. Tate and A.E. Smith, "Unequal Area Facility Layout Using Genetic Search", IEE Transactions, Vol 27, pp.465–472, 1995.

[31] R. Tavakkoli-Moghaddain and E. Shayan, "Facilities Layout Design by Genetic Algorithms". Computers & Industrial Engineering, Vol 35, pp. 527-530, 1998.

[32] J.A. Tompkins, J.A. White, Y.A. Bozer and J.M.A. Tanchoco, Facilities Planning, Wiley, 3rd ed, New York, 2003.

[33] M.J. Wang, M.H. Hub and M.Y. Ku, "A Solution to the Unequal Area Facilities Layout Problem by Genetic Algorithm", Computers in Industry, Vol 56, pp.207–220, 2005.

[34] Y. Wu and E. Appelton, "The Optimisation of Block Layout and Aisle Structure by a Genetic Algorithm". Computers & Industrial Engineering, Vol 41, pp. 371-387, 2002.