The Generalised Machine Layout Problem

Juan Jaramillo
Farmingdale State University of New York, jaramijr@farmingdale.edu

Alan McKendall
West Virginia University, alan.mckendall@mail.wvu.edu

International Journal of Production Research

ISSN: 0020-7543 (Print) 1366-588X (Online) Journal homepage: https://www.tandfonline.com/loi/tprs20

The generalised machine layout problem

J.R. Jaramillo & A.R. McKendall Jr

International Journal of Production Research
Vol. 48, No. 16, 15 August 2010, 4845–4859

The generalised machine layout problem

J.R. Jaramillo[a]* and A.R. McKendall Jr[b]

[a]*Department of Business Administration, Albany State University, 504 College Drive, Albany, GA 31705, USA;* [b]*Department of Industrial and Management Systems Engineering, West Virginia University, 325A Mineral Resources Building, Morgantown, WV 26506, USA*

(*Received 18 December 2008; final version received 10 June 2009*)

The Generalised MAchine Layout Problem (GMALP) is a generalisation of the integrated machine and layout problem, which is an extension of the machine layout problem. More specifically, the GMALP is the designing of a facility layout by defining the product mix, selecting the number of machines to be used, assigning these machines to the plant floor, and assigning products to machines such that total profit is maximised. Moreover, the GMALP integrates the quadratic assignment problem with a multicommodity flow problem. Therefore, the GMALP is a computationally intractable problem. Consequently, a mixed-integer nonlinear programming model was developed and used to solve small problem instances. Also, two simple construction algorithms and a tabu search (TS) heuristic were developed for solving large GMALP instances in acceptable computation times. In addition, a test dataset was used to evaluate the performances of the TS heuristic using the different construction algorithms. The results show that the TS heuristic perform slightly better with the second construction algorithm.

Keywords: optimisation; operational research; meta-heuristics; logistics; facility layout; supply chain management

1. Introduction

The problem of assigning machines to locations in a facility (i.e. a manufacturing plant) such that material handling cost (MHC) is minimised is known as the machine layout problem (MLP). Moreover, the MLP commonly assumes that flow amounts between machines are known beforehand, that the plant floor is represented as an array of equal size grid units, and that the plant floor has enough capacity to allocate all machines. Therefore, the MLP can be modeled as a Quadratic Assignment Problem (QAP). The QAP was introduced by Koopmans and Beckmann (1957), and was proven NP Hard by Sahni and Gonzales (1976). For an extensive review of solution techniques for the QAP, refer to Burkard *et al.* (1998) and Loiola *et al.* (2007).

An extended version of the MLP includes machine replicas (i.e. more than one machine of the same type). Considering machine replicas implies that flows between machines assigned to locations become part of the problem output, since static flows restrict the problem too much. Notice that the extended MLP can be modeled as a combination of the QAP and a multicommodity flow problem (MFP). As a consequence, MLP with machine

*Corresponding author. Email: juan.jaramillo@asurams.edu

ISSN 0020–7543 print/ISSN 1366–588X online
© 2010 Taylor & Francis
DOI: 10.1080/00207540903117840
http://www.informaworld.com

replicas is a much more complex problem. There are only a few papers in the literature that consider versions of the extended MLP. They are the integrated machine allocation and layout problem (IMALP) presented by Urban *et al*. (2000) and Jaramillo and McKendall (2009), an extended distance-based facility layout problem (EDFL) presented by Castillo and Peters (2003), and the dynamic extended facility layout problem (DEFLP) presented by Jaramillo and McKendall (2004). The IMALP and the EDFL start with a given set of machines (including machine replicas), and assume that total machine capacity is enough to completely satisfy all product demands. Therefore, the objective function of both problems is to minimise total MHC. The major difference between the IMALP and the EDFL is that the latter also considers grouping machines to form departments. Finally, the DEFLP expands the scope of the EDFL by adding more issues to the facility design problem, and by considering layout rearrangements according to a multi-period planning horizon (forecasted demand changes).

A realistic generalisation of the extended MLP is to include machine selection as part of the problem output. In both the IMALP and the EDFL the total capacity of the selected machines is enough to satisfy product demands. However, it is possible that the total capacity of selected machines is not enough to satisfy product demands. If that is the case, solutions (outputs) should favour the most profitable product mix. Therefore, some product demands will be partially or not satisfied at all. Favouring the most profitable product mix sometimes implies not producing certain products when they are not profitable. Grouping all these issues into one problem leads to the so-called Generalised MAchine Layout Problem (GMALP), which is the topic of this paper. The GMALP differs from the IMALP and the EDFL, since the additional problem, of determining which products to produce such that profit is maximised, is considered. This is the first paper, known to the authors, which considers this version of the extended MLP.

The applications of the GMALP are in the designing of manufacturing production systems. Particularly, the GMALP is useful in cases in which product demands are larger than the manufacturing production systems' capacities. In these cases, the GMALP allows the designer to find the most profitable product mix, to select only indispensable machines, and to allocate these machines in the best manner. In addition, the GMALP may exclude products that do not generate profits from the production mix, even when there is enough capacity to produce them. Commonly, unsatisfied demand can be satisfied by outsourcing product manufacturing or increasing production in other facilities.

The objectives of this paper are to introduce the GMALP to the literature, and to present a mathematical formulation, two simple construction algorithms, and a tabu search (TS) heuristic for the GMALP. The remainder of this paper is structured as follows. Section 2 describes the GMALP in detail as well as its mathematical model. Section 3 presents two simple construction algorithms and a TS heuristic for the GMALP, and Section 4 discusses the experimental results of the proposed heuristics. Finally, Section 5 summarises the findings of this research.

2. The generalised machine layout problem

Formally, the GMALP can be defined as the designing of a manufacturing layout by simultaneously defining a set of machines to process products, allocating the machines to the plant floor, selecting a product mix, and assigning product flows between machines such that profit is maximised. Profit is obtained by subtracting MHC and machine installation costs from product revenues. Product revenue is defined as the value of a product. The inputs of the problem are product demands, product routes, product capacity requirements, MHC per product unit per unit of distance, machine capacities, and plant floor details (e.g., distances between locations on the plant floor). The outputs of the problem are a set of products to produce, a set of machines (including machine replicas), machine locations on the plant floor, product flows between machines, and total profits. Finally, the assumptions for the GMALP considered in this research are as follows.

(a) The plant floor is represented as an array of equal size grid units, and each grid unit represents an available location. Distances between locations are known beforehand.
(b) Only one machine can be assigned to each location, and any machine can be assigned to any location.
(c) Product routes, requirements, demands, MHC per product unit per unit of distance, and product revenues are given.
(d) No alternative routes are allowed. Also, each operation can be performed by only one machine type.
(e) A product route does not visit the same machine type more than once, and each product route considers at least two machine types.
(f) Machine costs consist of purchasing and installation costs, and do not change from location to location.
(g) Machine capacities are given.
(h) The number of identical machines is limited based on the number of locations available on the plant floor.

Assumptions (a) and (b) are frequently used in MLP. Assumption (c) deals with product information. Assumptions (d) and (e) restrict product routes; these assumptions can be relaxed for more complex versions of the problem. Assumption (f) is commonly used in MLP, QAP, and facility layout problems and can easily be relaxed. In assumption (g), machine capacities are determined based on the amount of time machines are available (uptime). In assumption (h), the number of identical machines is limited based on the number of locations available on the plant floor. However, the number and type of machines purchased is also limited based on the objective of selecting machines that maximises profits (i.e. expensive machines used to produce only products that result in very little revenue would not be purchased).

The following example illustrates the GMALP and its assumptions. Tables 1 and 2 and Figure 1 summarise a problem instance. The instance considers three different products. The first column of Table 1 includes product routes (i.e. machine processing order). Columns 2, 3, and 4 contain machine processing times. Column 5 shows product demands in units. Column 6 is product revenues. Column 7 is MHC per product unit per distance unit. Row 6 shows the total machine processing time, in time units, required to satisfy all product demands. Row 7 is the capacity of each machine type in time units. Row 8 shows machine investment cost per machine. Row 9 shows the minimum number of machines of each type required to satisfy product demands completely. In addition, Figure 1 contains the plant floor layout. Notice that there are six locations available, and total demand satisfaction requires seven machines (i.e. $2+3+2$). Therefore, it is not possible to satisfy all product demands. Finally, Table 2 shows distances between locations.

Table 1. Product information.

	Machine requirements					
Product (route)	1	2	3	Demand	Revenue	MHC
1 (3–2–1)	0.13	0.57	0.67	4,500	\$50	\$5
2 (2–1–3)	0.60	0.24	0.10	8,100	\$38	\$3
3 (1–2–3)	0.56	0.83	0.39	12,000	\$87	\$8
Total requirements	12,165	14,469	8,505			
Machine capacities	7,100	7,000	6,500			
Machine cost (1000)	\$156	\$128	\$112			
Required machines	2	3	2			

Table 2. Distances between locations.

	L1	L2	L3	L4	L5	L6
L1	–	1	2	1	2	3
L2	1	–	1	2	1	2
L3	2	1	–	3	2	1
L4	1	2	3	–	1	2
L5	2	1	2	1	–	1
L6	3	2	1	2	1	–

L1	L2	L3
L4	L5	L6

Figure 1. Plant floor.

M3	M2	M3
	M1	M2

Figure 2. A solution layout.

A possible solution of the problem instance is shown in Figure 2 and Table 3. Figure 2 is the machine layout. Notice that two machines of type 3 are assigned to locations 1 and 3, respectively, and location 4 is left empty. Table 3 includes machine investment cost, product revenues, MHC, profit, and flows between machines.

2.1 *Mathematical formulation*

The following is the notation and the mixed-integer nonlinear programming formulation for the GMALP.

Notation:

p	product: $p = 1, \dots, P$
i, j	location: $i, j = 1, \dots, N$

Table 3. Solution summary.

Machine investment:	\$636,000.0	Prod. revenues:	\$1,237,151.8
Material handling cost:	\$231,142.9	Profit:	\$370,008.9

Product flow details

Product	Origin	Destination	Flow
1	M3 (L1)	M2 (L2)	4,500
	M2 (L2)	M1 (L5)	4,500
3	M1 (L5)	M2 (L2)	5,343.37
	M1 (L5)	M2 (L6)	5,290.56
	M2 (L2)	M3 (L1)	5,343.37
	M2 (L6)	M3 (L3)	5,290.56

m	machine type: $m = 1, \dots, M$
o	operation: $o = 1, \dots, O_p$, where O_p is the last operations required by product p
d_{ij}	distance from location i to location j
Dem_p	demand of product p
Rev_p	revenue from producing one unit of product p
c_p	cost of moving one unit of product p one distance unit
C_m	capacity of machine type m (in time units)
T_{pom}	$= \begin{cases} 1, & \text{if part } p \text{ requires machine } m \text{ at operation } o \\ 0, & \text{otherwise} \end{cases}$
I_m	installation cost of machine m
r_{po}	processing requirements of product p at operation o in time units per product unit
x_{ij}^p	flow between locations i and j of product p
y_{mi}	$= \begin{cases} 1, & \text{if a machine of type } m \text{ is assigned to location } i \\ 0, & \text{otherwise} \end{cases}$

Mathematical model:

$$\text{Maximise} \quad \sum_{p=1}^{P}\sum_{m=1}^{M}\sum_{i=1}^{N}\sum_{j=1}^{N} Rev_p T_{p1m} x_{ij}^p y_{mi} - \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{p=1}^{P} d_{ij} c_p x_{ij}^p - \sum_{m=1}^{M}\sum_{i=1}^{N} I_m y_{mi}, \tag{1}$$

subject to

$$\sum_{m=1}^{M} y_{mi} \le 1, \quad \forall i, \tag{2}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{m=1}^{M} T_{p1m} x_{ij}^p y_{mi} \le Dem_p, \quad \forall p, \tag{3}$$

$$\sum_{j=1}^{N}\sum_{p=1}^{P}\sum_{o=1}^{O_p-1} r_{po} T_{pom} x_{ij}^p y_{mi} + \sum_{p=1}^{P}\sum_{j=1}^{N} r_{pO_p} T_{pO_pm} x_{ji}^p y_{mi} \le C_m y_{mi}, \quad \forall i, m, \tag{4}$$

$$\sum_{m=1}^{M} T_{p(o-1)m} x_{ij}^{p} y_{mi} = \sum_{m=1}^{M} T_{pom} x_{ij}^{p} y_{mj}, \quad \forall p, i, j, o = 2, \ldots, O_p, \tag{5}$$

$$\sum_{i=1}^{N} \sum_{m=1}^{M} T_{p(o-1)m} x_{ij}^{p} y_{mi} = \sum_{i=1}^{N} \sum_{m=1}^{M} T_{p(o+1)m} x_{ji}^{p} y_{mi}, \quad \forall p, j, o = 2, \ldots, O_p - 1, \tag{6}$$

$$x_{ij}^{p} \geq 0, \quad \forall p, i, j, \tag{7}$$

$$y_{mi} \in [0, 1], \quad \forall m, i \tag{8}$$

Objective function (1) maximises total profit. Notice that the first term in the objective function gives total revenue, the second gives total MHC, and the third gives total machine investment cost. Constraint set (2) ensures that at most one machine is assigned to each location. Constraint set (3) ensures that product demands are not exceeded. Constraint set (4) guarantees that machine capacities are not exceeded. Constraint set (5) ensures that product flows follow their respective routes. Constraint set (6) ensures flow conservation at each location. Constraint sets (7) and (8) restrict the decision variables. It is important to mention that the GMALP (i.e. constraint set (2)) allows empty locations. As a result, the GMALP considers both cases: where space may be limited (i.e. the number of machines required to satisfy total demand exceeds the number of available locations) and where space may not be limited (i.e. there are sufficient locations to assign all the machines required to satisfy total demand).

Notice that the above mathematical formulation is nonlinear. However, the formulation can be linearised by replacing the product $x_{ij}^{p} y_{mi}$ with w_{ijm}^{p}, which yields the following objective function and additional constraints:

$$\text{Maximise} \quad \sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{i=1}^{N} \sum_{j=1}^{N} Rev_p T_{p1m} w_{ijm}^{p} - \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{p=1}^{P} d_{ij} c_p x_{ij}^{p} - \sum_{m=1}^{M} \sum_{i=1}^{N} I_m y_{mi}, \tag{9}$$

$$w_{ijm}^{p} - Dem_p y_{mi} \leq 0, \quad \forall p, i, j, m, \tag{10}$$

$$w_{ijm}^{p} - x_{ij}^{p} \leq 0, \quad \forall p, i, j, m, \tag{11}$$

$$x_{ij}^{p} - w_{ijm}^{p} + Dem_p y_{mi} \leq Dem_p, \quad \forall p, i, j, m, \tag{12}$$

$$w_{ijm}^{p} \geq 0, \quad \forall p, i, j, m. \tag{13}$$

3. Approximation algorithms

Since the GMALP is a computationally intractable problem, only small problem instances can be solved in a reasonable time using the above mixed-integer linear programming formulation (i.e. objective function (9) subject to constraints (2)–(13)). Consequently, approximation techniques are required for solving large size problems. Two types of approximation techniques are presented in this section, construction algorithms and a TS heuristic. First, a solution representation is presented for the GMALP.

3.1 *Solution representation*

A GMALP instance solution consists of a machine layout and a set of flows between machines. The machine layout can be represented as a solution vector $\boldsymbol{S} = \{s(1), s(2), \ldots, s(i), \ldots, s(N)\}$. Therefore, each vector position i represents a location, and a position value $s(i)$ represents a machine type assigned to location i. For example, the machine layout shown in Figure 2 can be represented as $\boldsymbol{S} = \{3, 2, 3, 0, 1, 2\}$. Notice that $s(4) = 0$ indicates that location 4 is empty. Assigning machines to locations is equivalent to assigning binary values to variables y_{mi} in the above formulation (i.e. $y_{31} = y_{22} = y_{33} = y_{04} = y_{15} = y_{26} = 1$ and all other $y_{mi} = 0$ for $\boldsymbol{S}$ defined above). Therefore, the remaining part of the problem, assigning products to machines (i.e. multicommodity flow problem (MFP)), can be solved optimally using a linear programming formulation and a linear programming technique such as simplex or an interior point method.

The number of possible layouts associated with the GMALP is much larger than the number of layouts of a QAP with the same number of locations. For example, for a QAP with six locations there are $6! = 720$ possible layouts. For the GMALP instance considered above, the number of possible layout solutions is given by

$$\text{layouts} = (\text{machine\_types} + 1)^{\text{locations}}.$$

That is, the number of possible layouts associated with the above instance is 4096. Notice that an empty layout is a possible solution. Moreover, for each possible layout, there is a corresponding MFP. Therefore, the GMALP is much more complex than the QAP, and requires a larger amount of computational time to solve. As a consequence, restricting the solution space of the problem reduces computational time. For the GMALP, the solution space can be reduced by restricting the number of machines to consider according to demand requirements (i.e. minimum number of machines that satisfies the demand). Notice that limiting the number of machines based on meeting demand requirements may produce inferior solutions, since the solution space is reduced such that the corresponding solution space may not contain the optimal solution. For example, the last row of Table 1 shows that seven machines are required to satisfy product demand. The reduced set of machines is called the required machine set defined as $\boldsymbol{M}$. For the above example, $\boldsymbol{M} = \{1, 1, 2, 2, 2, 3, 3\}$. Notice that any layout is a subset of the set $\boldsymbol{M}$ (since $|\boldsymbol{M}| \geq$ number of locations $= N$ where $|\boldsymbol{M}|$ is the cardinality of the set $\boldsymbol{M}$). Therefore, the solution can be represented as two sets ($\boldsymbol{S}$ and $\boldsymbol{S}'$) such that $\boldsymbol{S} \cup \boldsymbol{S}' = \boldsymbol{M}$. Then, the solution shown in Figure 2 can be represented as $\boldsymbol{S} = \{3, 2, 3, 0, 1, 2\}$ and $\boldsymbol{S}' = \{1, 2\}$, where $\boldsymbol{S}'$ gives the machines in $\boldsymbol{M}$ not assigned to locations. Finally, recall that, for each solution $\boldsymbol{S}$ with $\boldsymbol{S}'$ obtained, there is an MFP to be solved, which is discussed below.

Generating and solving many linear programs (MFPs) is computationally expensive. Therefore, in order to reduce computation time, a dummy location (L_{N+1}) with unlimited capacity is used to store machines in $\boldsymbol{S}'$. L_{N+1} allows the generation of a linear program including all machines in $\boldsymbol{M}$. Figure 3 shows a layout solution including L_{N+1} (i.e. L7). Therefore, L7 stores machines in $\boldsymbol{S}'$. In addition, Figure 4 illustrates the MFP associated with $\boldsymbol{S}$ and $\boldsymbol{S}'$ (each product network is shown independently for clarity). Notice that each node represents a particular machine in $\boldsymbol{M}$. Also, node S and node D are source and

L1 **M3**	L2 **M2**	L3 **M3**
L4	L5 **M1**	L6 **M2**

L7 **M1** **M2**

Figure 3. Machine layout including dummy location.

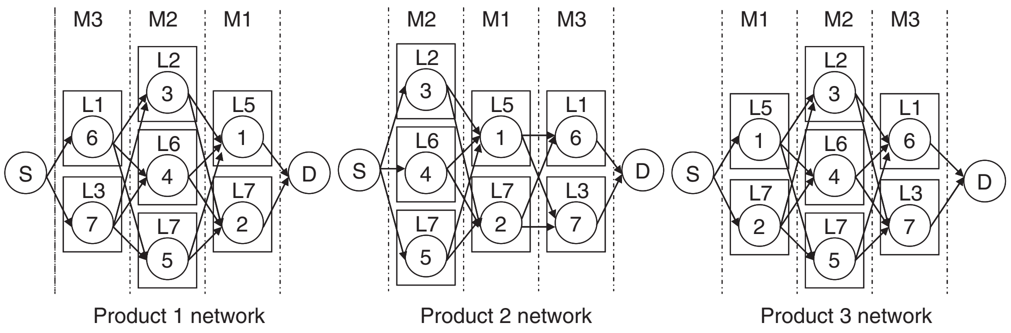

Figure 4. Product networks.

destination nodes added to facilitate the MFP formulation. For the above example, node 1 represents the first machine of type 1 (assigned to location 5), node 2 represents the second machine of type 1 (assigned to location 7), node 3 represents the first machine of type 2 (assigned to location 2), and so on. Arcs represent product routes, and arc costs represent MHC. As an example, for product 2 there is an arc connecting nodes 3 and 1 (i.e. a machine of type 2 and a machine of type 1). Since node 1 is at location 5 and node 3 is at location 2, the arc cost per product unit is obtained as product 2 MHC times the distance from location 5 to location 2 (i.e. $3*1 = $3 per product unit). In addition, distances between nodes connecting to nodes assigned to L_{N+1} are represented by large values, making these machines unattractive.

The following is the notation and the linear programming formulation for the MFP.

Notation:

k, l	nodes (machine numbers), $k, l = 0, \ldots, M+1$ where 0 is the source node, M is the total number of machines, and $M+1$ is the destination node
p	product, $p = 1, \ldots, P$, where P is the total number of products
Rev_p	revenue from producing one unit of product p
c_p	cost of moving one unit of product p one distance unit
d_{kl}	distance from node (machine number) k to node (machine number) l
b_k	capacity of machine number k
Dem_p	demand of product p
r_{pk}	processing requirements of product p at node (machine number) k
u_{kl}^{p}	$= \begin{cases} Dem_p, & \text{if node (machine number) } k \text{ immediately precedes node (machine number) } l \text{ according to product } p \text{ route} \\ 0, & \text{otherwise} \end{cases}$
x_{kl}^{p}	flow of product p from node (machine number) k to node (machine number) l

Mathematical model:

$$\text{maximise} \quad \sum_{p=1}^{P}\sum_{l=1}^{M} Rev_p x_{0l}^{p} - \sum_{p=1}^{P}\sum_{k=1}^{M}\sum_{l=1}^{M} c_p d_{kl} x_{kl}^{p}, \tag{1}$$

subject to

$$\sum_{l=1}^{M} x_{0l}^{p} \leq Dem_p, \quad \forall p, \tag{2}$$

$$\sum_{k=0}^{M} x_{kl}^{p} - \sum_{k=1}^{M+1} x_{lk}^{p} = 0, \quad \forall p;\, l = 1, \ldots, M, \tag{3}$$

$$\sum_{p=1}^{P}\sum_{l=1}^{M+1} r_{pk} x_{kl}^{p} \leq b_k, \quad k = 1, \ldots, M, \tag{4}$$

$$0 \leq x_{kl}^{p} \leq u_{kl}^{p}, \quad \forall\, p, k, l. \tag{5}$$

Objective function (1) maximises total profit by subtracting total MHC from total revenue. Constraint set (2) ensures that product demands are not exceeded. Constraint set (3) ensures flow conservation. Constraint set (4) guarantees that machine capacities are not exceeded. Constraint set (5) ensures that flows are non-negative and do not exceed upper bounds. Since the above formulation considers all machines in $\boldsymbol{M}$, constraint sets (2), (3), (4) and (5) are independent of machine locations. Moreover, the only difference among possible layouts is the distances between nodes (i.e. machine locations). Therefore, the same linear programming model with the same parameter settings (except the matrix d_{kl}) can be used for all possible machine layouts. In other words, only the parameter d_{kl} in the second term of the objective function needs to be updated before solving the formulation for each layout. More importantly, the optimal solution obtained for one layout can be used as a starting point when solving the formulation for another layout; therefore, the MFPs for all layouts after the first layout are not solved from scratch. Thus, computation time is reduced.

3.2 *Construction algorithms*

Construction algorithms provide initial solutions for improvement techniques. Two construction algorithms are presented in this paper, CA1 and CA2. CA1 randomly assigns machines to locations from the machine set $\boldsymbol{M}$. For example, if $\boldsymbol{M} = \{1, 1, 2, 2, 2, 3, 3, 4\}$ and there are six available locations. Then randomly pick a number between 1 and $8 = |\boldsymbol{M}|$. If, say, 3 is randomly selected, then assign the machine in the third position in set $\boldsymbol{M}$ (i.e. machine 2) to location 1 (i.e. $s(1) = 2$). Remove machine 2 in position 3 from $\boldsymbol{M}$, and randomly select a number between 1 and 7, and repeat this process until each location is assigned a machine. For instance, the solution $\boldsymbol{S} = \{2, 1, 3, 1, 2, 3\}$ may be obtained. Therefore, $\boldsymbol{S}' = \{2, 4\}$. The main limitation of CA1 is that it may generate layouts that may not process any product. That would be the case if all product routes visit a machine of type 4. Moreover, these kinds of layouts can cause problems for local search algorithms, since removing unused machines from the layout becomes more attractive than adding

Table 4. Moves performed on the current solution.

Current solution	$\boldsymbol{S} = \{3, 2, 3, 0, 1, 2\}$	$\boldsymbol{S}' = \{1, 2\}$
$\text{move}_{\text{swap}}$	$\boldsymbol{S}^N = \{\mathbf{2}, \mathbf{3}, 3, 0, 1, 2\}$	$\boldsymbol{S}' = \{1, 2\}$
$\text{move}_{\text{replace}}$	$\boldsymbol{S}^N = \{\mathbf{1}, 2, 3, 0, 1, 2\}$	$\boldsymbol{S}' = \{\mathbf{3}, 2\}$
$\text{move}_{\text{delete}}$	$\boldsymbol{S}^N = \{3, 2, 3, 0, \mathbf{0}, 2\}$	$\boldsymbol{S}' = \{\mathbf{1}, 1, 2\}$
$\text{move}_{\text{insert}}$	$\boldsymbol{S}^N = \{3, 2, 3, \mathbf{1}, 1, 2\}$	$\boldsymbol{S}' = \{2\}$
$\text{move}_{\text{jump}}$	$\boldsymbol{S}^N = \{3, 2, \mathbf{0}, \mathbf{3}, 1, 2\}$	$\boldsymbol{S}' = \{1, 2\}$

additional machines because of the machine installation cost. In order to overcome this drawback, CA2 was developed. CA2 considers maximising machine variety in the layout. CA2 iteratively assigns one machine of each type from $\boldsymbol{M}$ to $\boldsymbol{S}$ until all positions in $\boldsymbol{S}$ are filled. Then, the remaining machines are assigned to $\boldsymbol{S}'$. For the above example where $\boldsymbol{M} = \{1, 1, 2, 2, 2, 3, 3, 4\}$, the first iteration of CA2 leads to $\boldsymbol{S} = \{1, 2, 3, 4, \_, \_\}$ and $\boldsymbol{M} = \{1, 2, 2, 3\}$. After the second iteration $\boldsymbol{S} = \{1, 2, 3, 4, 1, 2\}$ is obtained. Since there is no more available locations in $\boldsymbol{S}$, the remaining machines are assigned to $\boldsymbol{S}' = \{2, 3\}$. Finally, product flows are obtained by solving the corresponding MFP optimally using the LP formulation defined above.

3.3 *Steepest ascent local search technique*

Local search (LS) techniques start with a solution $\boldsymbol{S}$ (i.e. the current solution) and explore neighbouring solutions with the hope of finding a better solution. More specifically, each neighbouring solution ($\boldsymbol{S}^N$) is obtained by performing a simple operation on the current solution $\boldsymbol{S}$. An operation performed on the current solution $\boldsymbol{S}$ is called a move, and the set of solutions that can be obtained by all possible moves is called the neighbourhood of solution $\boldsymbol{S}$ (i.e. $N(\boldsymbol{S})$). In this paper, the objective function value $f(\boldsymbol{S}^N)$ for each solution $\boldsymbol{S}^N \in N(\boldsymbol{S})$ is obtained, and the best solution $\boldsymbol{S}^N$ is defined as $\boldsymbol{S}^*$ (i.e. $f(\boldsymbol{S}^*) \geq f(\boldsymbol{S}^N)$ for each $\boldsymbol{S}^N \in N(\boldsymbol{S})$). If $f(\boldsymbol{S}^*) \geq f(\boldsymbol{S})$, then set $\boldsymbol{S} = \boldsymbol{S}^*$ and repeat the process. Otherwise, terminate the LS technique, since solution $\boldsymbol{S}$ is a local optimum. This technique is defined as a steepest ascent local search technique.

Initial solutions for the LS technique defined above can be obtained using CA1 and CA2. Also, recall that the GMALP may consider more machines than available locations ($|\boldsymbol{M}| \geq N$), and empty locations are allowed during the search process. As a consequence, five different moves are used to define the neighbourhood of solution $\boldsymbol{S}$. The first move, defined as $\text{move}_{\text{swap}}$, exchanges the locations of two machines that are in the layout $\boldsymbol{S}$. Notice that swapping two machines of the same type leads to the same solution, which is prohibited. Therefore, $\text{move}_{\text{swap}}$ is used only with machines of different type. The second move, defined as $\text{move}_{\text{replace}}$, exchanges the locations of a machine in $\boldsymbol{S}$ with a machine in $\boldsymbol{S}'$. The third move, defined as $\text{move}_{\text{delete}}$, transfers a machine from $\boldsymbol{S}$ to $\boldsymbol{S}'$. The fourth move, defined as $\text{move}_{\text{insert}}$, inserts a machine from $\boldsymbol{S}'$ into an empty location in $\boldsymbol{S}$. Finally, the fifth move, defined as $\text{move}_{\text{jump}}$, takes a machine already in $\boldsymbol{S}$ and reassigns it to an empty location. These moves are illustrated in Table 4.

3.4 *Objective function update*

As previously mentioned, one MFP is generated at the beginning of the search and is updated accordingly. Despite the reduction in computational time by solving the MFP starting from the last optimal solution, the process is still computationally expensive. As an alternative, Urban *et al.* (2000) and Castillo and Peters (2003) presented a technique that estimates $f(\boldsymbol{S}^*)$ without solving the MFP formulation optimally. For each move, the estimation technique keeps product flows between machines fixed and estimates $f(\boldsymbol{S}^*)$ based on new machine locations. Notice that the estimation technique only works for $\text{move}_{\text{swap}}$ and $\text{move}_{\text{jump}}$ moves. In contrast, any move associated with assigning a new machine to/from $\boldsymbol{S}'$ requires solving the MFP formulation since machines entering the layout have no product assignments, and flows assigned to deleted machines need to be reassigned. The main advantage of the estimation method is that it is computationally cheaper than solving the MFP optimally. However, its main disadvantage is that it may be misleading, guiding the search to select solutions that are not among the best candidates. However, this may allow for more diverse solutions. It is important to reiterate that the estimation technique is used for $\text{move}_{\text{swap}}$ and $\text{move}_{\text{jump}}$ moves. For a detailed explanation of the estimation technique, the reader is referred to Urban *et al.* (2000) and Castillo and Peters (2003).

3.5 *Tabu search algorithm*

Glover (1986) introduced the TS heuristic. Also, TS is described in detail by Glover (1989, 1990). Moreover, TS has been applied successfully to the QAP. Among the TS algorithms for the QAP are the simple TS of Skorin-Kapov, (1990), the robust TS of Taillard (1991), the reactive TS of Battiti and Tecchiolli (1994), and the TS with mutation operators for solution diversification presented by Misevicius (2005).

TS drives the LS technique by adding different features, allowing the search to escape from poor local optima. The main features of TS are the tabu list (*TList*), the aspiration criterion, and the stopping criterion. In addition to these, a new component, called x, is added to TS, which will be discussed below. *TList* keeps track of recent moves and makes them tabu for a certain number of iterations (i.e. tenure length, *TL*). The main use of *TList* is to lead the search to different and promising areas of the solution space and to avoid cycling. *TList* keeps track of the recent assignments of machine types to locations including not assigning machines to specific locations. *TList* is illustrated with two examples from Table 4. First, after $\text{move}_{\text{swap}}$, a machine of type 3 cannot visit location 1 and a machine of type 2 cannot visit location 2 for the next *TL* iterations. Second, after $\text{move}_{\text{insert}}$, location 4 cannot become empty in the next *TL* iterations. Finally, notice that there are no restrictions for machines entering or leaving the dummy location.

The aspiration criterion allows the acceptance of a tabu move when the move gives the best solution ever found ($\boldsymbol{S}^{\text{best}}$). Also, the stopping criterion determines the length of the search. The stopping criterion is to run TS for a certain amount of computation time (*StopTime*). Finally, the component x is used to speed up the search by using the estimation method discussed above when low-quality solutions are obtained. Once the search arrives at a promising area, $f(\boldsymbol{S}^N)$ is obtained by solving MFP optimally. More exactly, the estimation technique is used when $f(\boldsymbol{S}^{\text{best}}) - f(\boldsymbol{S}) \geq x$.

The TS heuristic is given as follows.

```
Step 1: Start TS
        Initialise components and parameters: TList, TL, ClockTime, StopTime, and $x$.
Step 2: Obtain an initial solution, $S^0$, using CA1 or CA2.
            Set $S^{best} = S^0$, and set $f(S^{best}) = f(S^0)$.
            Set $S = S^0$, and set $f(S) = f(S^0)$.
```

```
Step 3: Evaluate all possible moves for solution $S$ with $S'$ using the moves defined above
        (i.e. $move_{swap}$, $move_{replace}$, $move_{delete}$, $move_{insert}$, $move_{jump}$):
            If $f(S^{best}) - f(S) \geq x$ and move is either $move_{swap}$ or $move_{jump}$
                $f(S^N)$ is obtained using the estimation technique as discussed above.
            Else
                Solve MFP optimally to find $f(S^N)$ of each move.
        Select best admissible move (defined as move*), and perform move* on $S$ to obtain
        new solution $S^*$. The solution $S^*$ is obtained from the best admissible move,
        which is defined as the best non-tabu move or tabu move that overrides the tabu
        restriction.
            If $f(S^*) > f(S^{best})$
                Update $f(S^{best}) = f(S^*)$, and set $S^{best} = S^*$.
            Set $S = S^*$, and set $f(S) = f(S^*)$.
Step 4: Update TList as explained above.
Step 5: If $ClockTime \leq StopTime$
            Go to Step 3.
        Else
            Terminate the search.
```

4. Computational results

Since the GMALP is new in the literature, a dataset of 24 problems was developed in order to test the performance of the heuristics. (Note: the dataset is available upon request from the corresponding author.) The number of locations (N) considered are 6, 8, 9, 12, 15, 20, 25, and 30. In addition, three levels of product types were considered (i.e. three, six, and nine product types) for each layout. The GMALP formulation was solved using the CPLEX solver (version 6.6). Additionally, all heuristics were coded in Visual Basic 2005. Moreover, all problems were solved using a computer equipped with a 2.2 GHz AMD Turion 64 processor, 1 GB of memory, and windows XP. The solver used for solving the MFP formulation was lp_solve (version 5.5.0.10). lp_solve is an open-source (mixed-integer) linear programming system developed by Berkelaar *et al.* (2007).

Heuristic parameter settings were obtained through experimentation and they were set as follows. TL was set as a function of the number of positions ($\#p$) in the tabu list (i.e. (number of machine types + 1)*$N = \#p$). Three different TL values were tested in this research: $A = 0.05*\#p$ (i.e. TSA), $B = 0.10*\#p$ (i.e. TSB), and $C = 0.15*\#p$ (i.e. TSC). Moreover, it was observed through experimentation that the lower TL values often led to cycling, and larger TL did not provide competitive solutions in acceptable computational times. The parameter x was set in such a way that MFP was solved optimally for approximately 50% of the visited solutions. For the other 50%, $f(S^N)$ was estimated using the estimation technique. Recall that the main idea behind x is to solve only the MFP optimally using the formulation for promising solutions. It was observed that low values of x (i.e. solving more MFP formulations optimally) do not provide good solutions in acceptable computational times. This happens because the algorithm uses its computational time in solving MFP formulations for low-quality solutions (i.e. low $f(S^N)$). On the other hand, large x values (i.e. solving less MFP formulations optimally) produced inferior results. Solving less MFP formulations may miss some promising solutions (i.e. high $f(S^N)$). Recall that the estimation technique can be misleading. However, it may provide more diverse solutions. Moreover, using the estimation technique too aggressively (i.e. large x values) may lead the search into poor areas of the solution space.

As mentioned before, initial solutions can be obtained using CA1 and CA2. Recall that CA1 is a random algorithm that may generate different initial solutions for a given problem instance. On the contrary, CA2 is a deterministic algorithm that generates a unique solution for each problem instance. Therefore, it is possible to have two different variations of TS based on CA1 and CA2. The first, TS1, obtains initial solutions using CA1. The second, TS2, uses CA2 in order to generate starting solutions. Since CA1 may generate different initial solutions, CA1 can be used as a diversification strategy. Hence, five initial solutions are generated using CA1. Notice that combining TS1 and TS2 with the three different *TL* settings (i.e. TSA, TSB, and TSC), six different variations of TS are obtained (i.e. TS1A, TS1B, TS1C, TS2A, TS2B, and TS2C).

Finally, the stopping criterion of both algorithms was set as the heuristic run time. For a given instance, the computational time for one run of TS2 (i.e. TS2A, TS2B, and TS2C) equals the computational time of five runs of TS1 (i.e. TS1A, TS1B, and TS1C).

Table 5 gives a summary of the results obtained using the linearised GMALP formulation introduced above, and the tabu search algorithm with both construction algorithms. The first column of Table 5 is the problem identification number. The second column is the TS stopping criterion (i.e. computational time in minutes). Additionally, columns 3–5 show the best objective function values (OFVs) obtained

Table 5. Summary of results.

	Time	TS1			TS2		
Id #	(min)	*A*	*B*	*C*	*A*	*B*	*C*
1	0.05	**479,138.8***	**479,138.8***	**479,138.8***	**479,138.8***	**479,138.8***	**479,138.8***
2	0.05	**831,825.7***	**831,825.7***	**831,825.7***	**831,825.7***	**831,825.7***	**831,825.7***
3	0.1	**714,113.2***	**714,113.2***	**714,113.2***	**714,113.2***	**714,113.2***	**714,113.2***
4	0.1	**300,224.3***	**300,224.3***	**300,224.3***	**300,224.3***	**300,224.3***	**300,224.3***
5	0.1	**873,724.1***	**873,724.1***	**873,724.1***	**873,724.1***	**873,724.1***	**873,724.1***
6	0.15	**287,515.5**	**287,515.5**	**287,515.5**	**287,515.5**	**287,515.5**	**287,515.5**
7	0.1	**1,679,830.0***	**1,679,830.0***	**1,679,830.0***	**1,679,830.0***	**1,679,830.0***	**1,679,830.0***
8	0.3	**699,745.1**	**699,745.1**	**699,745.1**	**699,745.1**	**699,745.1**	**699,745.1**
9	0.5	**265,666.4**	**265,666.4**	**265,666.4**	**265,666.4**	**265,666.4**	**265,666.4**
10	1	**978,611.8**	**978,611.8**	**978,611.8**	**978,611.8**	**978,611.8**	**978,611.8**
11	2	**1,423,330.6**	**1,423,330.6**	**1,423,330.6**	**1,423,330.6**	**1,423,330.6**	**1,423,330.6**
12	3	**1,094,023.7**	**1,094,023.7**	**1,094,023.7**	**1,094,023.7**	**1,094,023.7**	**1,094,023.7**
13	2	**2,320,169.9**	**2,320,169.9**	**2,320,169.9**	**2,320,169.9**	**2,320,169.9**	**2,320,169.9**
14	10	**846,341.6**	**846,341.6**	832,076.6	**846,341.6**	**846,341.6**	**846,341.6**
15	15	**459,980.3**	**459,980.3**	454,047.9	**459,980.3**	**459,980.3**	**459,980.3**
16	20	**1,359,022.5**	**1,359,022.5**	1,351,609.9	1,351,609.9	**1,359,022.5**	**1,359,022.5**
17	40	2,470,843.6	**2,486,778.5**	2,402,827.6	2,470,843.6	2,434,944.0	2,435,290.8
18	80	1,839,527.4	1,844,075.9	1,827,421.6	**1,895,744.7**	1,843,526.1	1,809,227.0
19	70	3,437,856.7	3,410,644.6	3,421,253.7	**3,442,248.5**	3,413,711.3	3,333,302.7
20	110	3,060,824.2	3,057,702.1	3,079,853.9	3,001,698.3	3,001,698.3	**3,080,906.4**
21	200	1,213,148.5	1,195,719.1	1,191,799.7	**1,220,033.9**	1,218,171.4	1,195,193.2
22	100	**2,072,452.3**	2,032,908.0	2,035,559.4	2,069,259.5	2,004,177.7	2,003,789.7
23	250	2,883,321.9	2,850,932.2	2,893,217.3	2,923,039.0	**2,927,297.0**	2,868,216.0
24	400	2,367,091.1	**2,394,338.4**	2,372,920.7	2,353,936.8	2,369,895.3	2,363,280.2

using TS1. Also, columns 6–8 are the best OFVs obtained using TS2. Furthermore, bold values in the table represent the best OFV found for each problem. Finally, bold values with an asterisk represent OFV from known optimal solutions.

Optimal solutions were found using the linearised GMALP formulation (i.e. objective function (9) subject to constraints (2)–(13)). Moreover, the formulation was used in problems 1–9. Optimal solutions were obtained for problem instances 1–5 and 7. However, the execution of the CPLEX solver was terminated for problems 6, 8, and 9 after 166 h of computational time. That is, the largest problem that could be solved optimally in a reasonable computational time considers nine locations and three products.

The results obtained for TS algorithms show that TS2 obtained the best solution in 21 instances while TS1 found the best solutions in 19 instances. Also, TS1A found the best solution in 17 opportunities, TS1B in 18, TS1C in 13, TS2A in 18, TS2B in 17, and TS2C in 17. When comparing TS1B and TS2A, it is observed that TS2A obtained better solutions in five instances, the same solutions in 15, and worse solutions in four instances. Hence, TS2 slightly outperformed TS1 for this dataset. As a result, when using TS, it is recommended to use CA2 and set $TL = 0.05{*}\#p$.

The tabu search algorithm presented in this research incorporates five different moves, as explained above. These moves allow the algorithm to cope with the complexity of the GMALP. Moreover, these moves allow the swapping of machine locations, moving machines to empty locations, deleting machines, and inserting new machines in order to evaluate new solutions. Also, the parameter x has a critical role in the efficiency and speed of TS. Recall that x is used to decide if MFPs are solved optimally or not. Notice that solving too many MFPs optimally makes TS too computationally expensive for practical purposes. On the other hand, not solving enough MFP optimally leads TS to poor solutions, since the algorithm misses promising areas of the solution space. That is, x provides the balance that makes TS effective and practical.

5. Conclusions

This paper introduces the GMALP, which is a new problem in the literature that provides a more realistic approach to MLP. Consequently, a mathematical formulation of the problem, two simple construction algorithms (CA1 and CA2), and a TS meta-heuristic, are presented. The results obtained show that TS with the deterministic construction algorithm (CA2) performed better than the TS with the random construction algorithm (CA1). For future research it is recommended to develop a strategy that solves the QAP and the MFP simultaneously. That is, an approach that changes layouts and updates product flow assignments in one step. Also, it is suggested to develop other meta-heuristics, such as hybrid ant systems and path relinking.

References

Battiti, T. and Tecchiolli, G., 1994. The reactive tabu search. *ORSA Journal of Computing*, 6 (2), 26–140.

Berkelaar, M., Eikland, K., and Notebaert, P., (2007). lp_solve website [online]. Source: Available from: http://www.geocities.com/lpsolve [Accessed 30 April 2008].

Burkard, R.E., *et al.*, (1998). *The quadratic assignment problem*. Karl-Francez University and Graz Technical University. Report 126.

Castillo, I. and Peters, B., 2003. An extended distance based facility layout problem. *International Journal of Production Research*, 41 (11), 2451–2479.

Glover, F., 1986. Future paths for inter programming and links to artificial intelligence. *Computers and Operations Research*, 1 (3), 533–549.

Glover, F., 1989. Tabu search—Part I. *ORSA Journal on Computing*, 1 (3), 190–206.

Glover, F., 1990. Tabu search—Part II. *ORSA Journal on Computing*, 2 (1), 4–32.

Jaramillo, J.R. and McKendall, A.R., (2004). Dynamic extended facility layout problem. *In*: *IERC annual conference*, Houston, TX, 6 pages on CDROM.

Jaramillo, J.R. and McKendall, A.R., 2009. Metaheuristics for the integrated machine allocation and layout problem. *International Journal of Operational Research*, forthcoming.

Koopmans, T.C. and Beckmann, M.J., 1957. Assignment problems and the location on economic activities. *Econometrica*, 25 (1), 53–76.

Loiola, L., *et al.*, 2007. A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176 (2), 657–690.

Misevicius, A., 2005. A tabu search algorithm for the quadratic assignment problem. *Computational Optimization and Applications*, 30 (1), 95–111.

Sahni, S. and Gonzales, T., 1976. P complete approximation problems. *Journal of the Association of Computing Machinery*, 23 (3), 555–565.

Skorin-Kapov, J., 1990. Tabu search applied to the quadratic assignment problem. *ORSA Journal of Computing*, 2 (1), 33–45.

Taillard, E., 1991. Robust tabu search for the quadratic assignment problem. *Parallel Computing*, 17 (4–5), 443–455.

Urban, T.L., Chiang, W.-C., and Rusell, T., 2000. The integrated machine allocation and layout problem. *International Journal of Production Research*, 38 (13), 2911–2930.