2010

Metaheuristics for the Integrated Machine Allocation and Layout Problem

Juan Jaramillo

Alan McKendall

Metaheuristics for the Integrated Machine Allocation and Layout Problem

Juan R. Jaramillo*

Department of Business Administration,
Albany State University,
504 College Drive, Albany, GA 31705, USA
Fax: +229 430 5119
E-mail: juan.jaramillo@asurams.edu
*Corresponding author

Alan R. McKendall

Department of Industrial and
Management Systems Engineering,
West Virginia University,
325A Mineral Resources Building,
Morgantown, WV 26505, USA
Fax: +304 293 4970
E-mail: alan.mckendall@mail.wvu.edu

Abstract: The Integrated Machine Allocation and Layout Problem (IMALP) is the problem of assigning a set of machines (including machine replicas) to locations while assigning product flows to machines such that Material Handling Cost is minimised. A new mathematical formulation, a Tabu Search (TS) heuristic, and a Memetic Algorithm (MA) are presented for the IMALP. The algorithms were evaluated using a set of test problems available in the literature. TS and the MA obtained equal or better solutions for the dataset than previous techniques presented in the literature. More specifically, TS obtained better solutions in 47% of the instances, and MA improved the best known solution in 52.4% of the cases. As a result, MA out-performed TS with respect to solution quality and computation time.

Keywords: Integrated Machine Allocation and Layout Problem; IMALP; Memetic Algorithms; MA; metaheuristics; multicommodity flow problem; quadratic assignment problem; Tabu Search; TS.

Reference to this paper should be made as follows: Jaramillo, J.R. and McKendall, A.R. (2010) 'Metaheuristics for the Integrated Machine Allocation and Layout Problem', *Int. J. Operational Research*, Vol. 7, No. 1, pp.74–89.

Biographical notes: Juan R. Jaramillo is an Assistant Professor in the Department of Business Administration at Albany State University. He received his PhD in Industrial Engineering, and his MS in Industrial Engineering at the West Virginia University. He received his BS in Civil Engineering and his BS in Geological Engineering at Escuela de Ingeniería de Antioquia in Colombia. His research is in developing efficient solution techniques in the area of logistics and supply chain. He has published in *Computers and Operations Research* and presented in the Institute of Operations Research (INFORMS) and the Institute of Industrial Engineers (IIE) annual conferences.

Alan R. McKendall is an Associate Professor in the Department of Industrial and Management Systems Engineering at the West Virginia University. He received his PhD in Industrial Engineering, MS in Industrial Engineering and MS in Applied Mathematics from the University of Missouri in Columbia. He received his BS degree in Mathematics from Southern University in New Orleans. His main research interest is in developing efficient algorithms in the areas of logistics and scheduling. He has published in journals such as *Computers and Operations Research*, *Information Sciences*, *Int. J. Industrial Engineering*, *Int. J.Operational Research* and *Int. J. Production Research*. Currently, he is on the Editorial Board of the *Open Operational Research Journal* and *Int. J. Mathematics in Operational Research*.

1 Introduction

The Integrated Machine Allocation and Layout Problem (IMALP) is the problem of assigning a given set of machines, including machine replicas, to locations on the plant floor while assigning product flows to machines such that Material Handling Cost (MHC) is minimised and product demands are satisfied. The problem of assigning machines to locations on the plant floor is defined as the Machine Layout Problem (MLP) which can be modelled as a Quadratic Assignment Problem (QAP). The QAP was first introduced by Koopmans and Beckmann (1957). The problem of assigning product flows to machines is defined as the Minimum Cost Multicommodity Flow Problem (MCMFP). Among the first to study multicommodity flow problems is Robacker (1955). As a result, the IMALP is the integration of two well-known problems: the QAP and the MCMFP. The IMALP was first introduced by Urban, Chiang and Rusell (2000). Following are the main assumptions of the IMALP as it was introduced in the literature and as it is treated in this research.

1 The plant floor is represented as an array of equal size grid units, and each grid unit represents an available location.

2 Distances between locations are known in advance.

3 Only one machine can be assigned to each location; and any machine can be assigned to any location without any additional cost.

4 Product routes, requirements and demands, are known in advance.

5 Each operation can be performed on only one machine type, and a product visits a machine type at most once.

6 The set of machines (including machine replicas) and their capacities are known in advance.

7 The set of machines equals the number of locations, and the number of machines is enough to satisfy all products demands.

Assumptions (1)–(3) and part of assumption 7 (i.e. the set of machines equals the number of locations) are the QAP assumptions. The rest of the assumptions are for the MCMFP.

The IMALP is illustrated using a simple example. Tables 1 and 2 and Figure 1 summarise the information of a small IMALP instance. Notice that the instance involves the production of three products. Products and product routes, which indicate the machine processing sequence, are shown in the first column of Table 1. Columns 2–5 give the machine requirements of each part in time units (e.g. each unit of product three requires 60 time units at a machine of type 2). Column 6 shows product demands. Row 6 gives machine type capacities (i.e. available time per machine). Row 7 gives the number of machines of each type that are available. Figure 1 shows the layout of the plant floor where there are six available locations to assign machines. Table 2 gives the distances between the locations on the plant floor. Assume MHC is \$1 per product unit per unit of distance for all parts. A possible solution for the IMALP instance is given in Figure 2 (i.e. machine layout) and Table 3 (i.e. product flows between machines assigned to locations). Figure 2 shows that a machine of type 3 is assigned to location 1, a machine of type 2 is assigned to location 2 and so on. Notice that the machine layout includes replicated machines (i.e. two machines of type 3, one at location 1 and another at location 6). In addition, Table 3 shows product flows between machines assigned to locations and their respective MHC. For example, second row shows that 20 units of product 1 travel from the machine of type 1 assigned to location 3 to the machine of type 2 at location 2 with a MHC of \$20. Similarly, row 8 shows that 23.33 units of product 3 travel from a the machine of type 1 at location 3 to the machine of type 2 at location 5 with a MHC of \$ 46.67. Finally, the Objective Function Value (OFV), sum of the MHC, for the solution is \$ 285.

Table 1 Product information

	Machine requirements [time units]				*Demand*
Product (route)	*1*	*2*	*3*	*4*	*[units]*
1 (1-2-3-4)	20	50	45	10	20
2 (1-3-4)	25	–	55	10	35
3 (1-2-4)	15	60	–	10	40
Machine capacity [time units]:	2,000	2,000	2,000	2,000	–
Available machines:	1	2	2	1	–

Table 2 Plant floor and 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	–

Table 3 Product flow solution details

Product	*Flow*	*From*	*To*	*MHC*
1	20	M1 (L3)	M2 (L2)	$20
	20	M2 (L2)	M3 (L1)	$20
	20	M3 (L1)	M4 (L4)	$20
2	35	M1 (L3)	M3 (L6)	$35
	35	M3 (L6)	M4 (L4)	$70
3	16.67	M1 (L3)	M2 (L2)	$16.67
	23.33	M1 (L3)	M2 (L5)	$46.67
	16.67	M2 (L2)	M4 (L4)	$33.33
	23.33	M2 (L5)	M4 (L4)	$23.33

Figure 1 Layout of plant floor

L1	L2	L3
L4	L5	L6

Figure 2 Layout of machines (solution)

L1 **M3**	L2 **M2**	L3 **M1**
L4 **M4**	L5 **M2**	L6 **M3**

As stated previously, the QAP is used to assign machines to locations on the plant floor, while the MCMFP assigns product flows between machines such that MHC is minimised. More importantly, the IMALP is a generalisation of the QAP. In fact, when flows between machines are part of the problem input, the IMALP reduces to the QAP. Moreover, since the QAP is NP hard (Sahni and Gozales, 1976) the IMALP is also NP hard (Urban, Chiang and Rusell, 2000). In addition, problems integrating layout and machine flow decisions are not common in the literature. Besides the IMALP, there are two other problems that integrate layout and product flow assignment decisions. These problems are the Extended Distance-based Facility Layout problem (EDFL) introduced by Castillo and Peters (2003), and the dynamic extended facility layout problem presented in Jaramillo and McKendall (2004). Another problem that is related, but considers only a single commodity (grain) is the problem of assigning silos to locations such that the sum of transportation and investment cost is minimised, which was presented by Foulds (2005). A major difference is that only one commodity is considered. For a recent review for the problem of assigning departments to locations, see Singh and Sharma (2006).

Considering that the IMALP is a computationally difficult problem, approximation techniques are required to find good solutions for medium and large size problem instances. It is important to mention that approximation techniques rely on evaluating a good amount of machine layouts. Consequently, for each machine layout, such as the one shown in Figure 2, there is a need to solve a corresponding MCMFP in order to find the optimal product flows between machines. Also, recall that MCMFP is a linear programming problem which can be solved in acceptable computational times by well-known techniques (i.e. simplex and interior point methods). In fact, flows shown in Table 2 were obtained by solving a MCMFP optimally for the layout shown in Figure 2. Moreover, in the MCMFP machines are represented by nodes, and paths between machines are represented by arcs. For a detailed description of MCMFP including a mathematical formulation, the reader is referred to Kennington (1978). Unfortunately, solving a large amount of MCMFP to optimality (i.e. one MCMFP for each possible machine layout) requires a large amount of computational time, making approximation algorithms for the IMALP computationally expensive. In addition, the number of constraints and variables of MCMFP increases rapidly with the size of the problem.

In order to solve the IMALP, Urban, Chiang and Rusell (2000) presented a Greedy Randomised Adapted Search Procedure (GRASP) metaheuristic that solves the problem iteratively. First, the algorithm generates an initial set of flows between machines. Then, GRASP is used to solve the MLP (QAP) while keeping flows between machines fixed. After solving the MLP (i.e. finding a 'good' machine layout), the MCMFP is solved again in order to find optimal product flows between machines for the 'good' machine layout. Then, GRASP is used with the new set of product flows. Finally, the procedure is repeated until no improvement is obtained when applying GRASP and solving the MCMFP consecutively. It is important to mention that Castillo and Peters (2003) applied a similar approach for the EDFL which uses simulated annealing instead of GRASP.

The purpose of this article is to introduce an alternative mathematical formulation, and to present a TS heuristic and a MA for the IMALP. The remaining of this work is as follows. Section 2 contains a new mixed integer linear programming model for the IMALP. Section 3 presents a TS heuristic and a MA. Section 4 discusses the results of the proposed heuristics obtained on a set of test problems taken from the literature. Finally, Section 5 summarises the findings of this work.

2 Mathematical formulation

Following is an alternative formulation for the IMALP. The alternative formulation incorporates the information about product routes. Also, the cubic term in the original IMALP formulation presented in Urban, Chiang and Rusell (2000) is replaced by a quadratic term. Therefore, the number of variables required to linearise the mathematical model is reduced. The following include the notation and formulation of the alternative mixed integer non-linear programming model for the IMALP.

Indexes:

p: Product, $p = 1, \ldots, P$.

i, j: Location, $i, j = 1, \ldots, N$.

m: Machine type: $m = 1, \ldots, M$.

o: Operation: $o = 1, \ldots, O_p$, where O_p is the number of operations required by product p.

Parameters:

d_{ij}: Distance from location i to location j.

Dem_p: Demand 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).

R_m: Machine type m replicas.

T_{pom}: 1 if part p requires machine of type m at operation o. 0 otherwise.

r_{po}: Processing requirements of product p at operation o in time units per product unit.

Variables:

x^p_{ij}: Flow from machine at location i to machine at location j of product p.

y_{mi}: 1 if a machine of type m is assigned to location i. 0 otherwise.

Mathematical model:

$$\text{Min} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{p=1}^{P} d_{ij} c_p x_{ij}^p \tag{1}$$

Subject to:

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

$$\sum_{i=1}^{N} y_{mi} = R_m \quad \forall m \tag{3}$$

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{m=1}^{M} T_{p,1,m} x_{ij}^p y_{mi} = \text{Dem}_p \quad \forall p \tag{4}$$

$$\sum_{j=1}^{N} \sum_{p=1}^{P} \sum_{o=1}^{O_p - 1} r_{po} T_{pom} x_{ij}^p y_{mi} + \sum_{p=1}^{P} \sum_{j=1}^{N} r_{pO_p} T_{pO_p m} x_{ji}^p y_{mi} \le C_m y_{mi} \quad \forall i, m \tag{5}$$

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

$$\sum_{i=1}^{N}\sum_{m=1}^{M} T_{p,o-1,m} x_{ij}^{p} y_{mi} = \sum_{i=1}^{N}\sum_{m=1}^{M} T_{p,o+1,m} x_{ij}^{p} y_{mi} \quad \forall p, j, o = 2, \ldots, O_p - 1 \tag{7}$$

$$x_{ij}^{p} \geq 0 \quad \forall i, j \tag{8}$$

$$y_{mi} \in [0,1] \quad \forall m, i \tag{9}$$

Objective function (1) minimises total MHC. Constraint set (2) ensures that only one machine is assigned to each location. Constraint set (3) ensures that the correct number of replicas of each machine type is assigned to the plant floor. Constraint set (4) ensures that product demands are met. Constraint set (5) guarantees that machine capacities are not exceeded. Constraint set (6) ensures that product flows follows their respective routes. Constraint set (7) ensures flow conservation. Constraint sets (8) and (9) restrict the decision variables. Notice that the mathematical formulation above is non-linear. However, the formulation can be linearised by adding a new variable w_{ijm}^{p} that replaces the product $x_{ij}^{p} y_{mi}$ and by including the following constraints:

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

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

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

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

Because of the large amount of variables and constraints generated by the above mathematical formulation, only small instances can be solved in acceptable computational times. Therefore, heuristics need to be developed to find good solutions for medium and large size problems.

3 Metaheuristics

As mentioned before, two metaheuristics, a TS and a MA, are presented for the IMALP. This section is divided in five subsections. Section 3.1 defines a solution representation for the IMALP. Section 3.2 introduces a steepest descent pairwise exchange heuristic. Construction algorithms are presented in Section 3.3. Sections 3.4 and 3.5 describe a TS heuristic and a MA, respectively.

3.1 Solution representation

Recall that the IMALP can be viewed as the combination of the QAP and the MCMFP. Therefore, the machine layout (i.e. QAP solution) can be represented as a permutation of machines or vector $\boldsymbol{S} = (s(1), \ldots, s(i), \ldots, s(N))$. Each position i in $\boldsymbol{S}$ represents a location, $s(i)$ represents a machine type assigned to location i, and N is the total number of locations. For example, the layout shown in Figure 2 can be represented as $\boldsymbol{S}$ = (3, 2, 1, 4,

2, 3). That is, a machine of type 3 is assigned to location 1 (i.e. $s(1) = 3$), a machine of type 2 is assigned to location 2 (i.e. $s(2) = 2$), and so on. Once machines have been assigned to the plant floor, the OFV, $f(\boldsymbol{S})$, and flows between machines/locations are obtained by solving the corresponding MCMFP optimally.

3.2 Construction algorithm

Construction algorithms are used to build initial solutions. Two construction algorithms are presented here. The first one, called CA1, assigns machines in order according to its type and availability. That is, starting with the set of available machines $\boldsymbol{M} = \{1, 2, 2, 3, 3. 4\}$ CA1 assigns one machine of each type from $\boldsymbol{M}$ to the first 4 locations (i.e. $s(1) = 1$, $s(2) = 2$, $s(3) = 3$, and $s(4) = 4$)). Then, the process is repeated for the machines remaining in $\boldsymbol{M}$ (i.e. $\boldsymbol{M} = \{2, 3\}$) until all machines are assigned (i.e. $s(5) = 2$ and $s(6) = 3$). Therefore, CA1 will generate the initial solution $\boldsymbol{S} = (1, 2, 3, 4, 1, 2)$. The second construction algorithm, CA2, randomly assign the machines in $\boldsymbol{M}$ to locations. An example of CA2 is the initial solution $\boldsymbol{S} = (3, 2, 3, 1, 4, 2)$.

3.3 Steepest descent Local Search technique

Local Search (LS) techniques have a fundamental role in metaheuristics such as TS and simulated annealing. These techniques start with an initial solution (i.e. current solution) and explore neighbouring solutions in order to find better solutions (i.e. to improve the current solution). The search technique used in this work is a steepest descent pairwise exchange LS technique. Consequently, LS explores neighbouring solutions by exchanging locations between pairs of machines of different type in the current solution $\boldsymbol{S}$. Each pairwise exchange is called a move. For example, using the solution shown in Figure 2 as the current solution $\boldsymbol{S} = (3, 2, 1, 4, 2, 3)$, a move is to exchange machine of type 2 in location 2 (i.e. $s(2) = 2$) with machine of type 1 at location 3 (i.e. $s(3) = 1$). After performing the move, the new solution is $\boldsymbol{S}^* = (3, \mathbf{1}, \mathbf{2}, 4, 2, 3)$. Notice that exchanging 2 machines of the same type do not generate a new layout (i.e. exchanging the machine of type 3 at location 1 with the machine of type 3 at location 6). Therefore, exchanging two machines of the same type is not considered as valid move. As a consequence there are 13 possible moves for $\boldsymbol{S}$.

After generating each new solution, it is necessary to obtain $f(\boldsymbol{S}^*)$. As it was mentioned before, $f(\boldsymbol{S}^*)$ can be obtained by solving the corresponding MCMFP. However, since at each iteration of the LS, a large amount of neighbouring solutions can be generated for large size problems, solving the MCMFP for each one of these layouts is computationally expensive. In fact, when problem instances increase in size (i.e. number of locations and number of products increases), the size of the MCMFP grows rapidly (i.e. number of variables and constraints increases). Alternatively, $f(\boldsymbol{S}^*)$ can be estimated by fixing flows between machines and updating $f(\boldsymbol{S})$ after each move accordingly. In fact, this alternative method is used in Urban, Chiang and Rusell (2000) and Castillo and Peters (2003), and the reader is referenced to these papers for a more detailed explanation. It is important to mention that the alternative method reduces the amount of computational time required to solve the corresponding MCMFP. On the other hand, solutions obtained with the alternative method are not guaranteed to be optimal. That is, the estimated OFV can be far from the optimal OFV. Table 4 shows the results obtained

for $f(\boldsymbol{S}^*)$ using the alternative method and solving the corresponding MCMFP optimally. Notice that the alternative method uses the same product flows as the current solution

Table 4 f(S*) Update comparison

			Alternative		*Optimal*	
Product	*From*	*To*	*Flow*	*MHC*	*Flow*	*MHC*
1	M1 (L2)	M2 (L3)	20	$20	20	$20
	M2 (L3)	M3 (L6)	20	$20	20	$20
	M3 (L6)	M4 (L4)	20	$40	20	$40
2	M1 (L2)	M3 (L1)	35	$35	35	$35
	M3 (L1)	M4 (L4)	35	$35	35	$35
3	M1 (L2)	M2 (L3)	16.67	$16.67	6.67	$6.67
	M1 (L2)	M2 (L5)	23.33	$23.33	33.33	$33.33
	M2 (L3)	M4 (L4)	16.67	$50	6.67	$20
	M2 (L5)	M4 (L4)	23.33	$23.33	33.33	$33.33

(see Table 3), therefore $f(\boldsymbol{S}^*) = \$263.33$. On the other hand, solving the MCMFP optimally reassign some of the product flows between machines leading to $f(\boldsymbol{S}^*) = \$243.33$. Moreover, $\boldsymbol{S}^*$ with the optimal set of flows between machines is the optimal solution of the simple example presented earlier. That was verified by solving the problem instance using the above mathematical formulation for the IMALP.

Last, LS selects the best neighbouring solution (i.e. $\boldsymbol{S}^{**}$) and makes it the new current solution (i.e. $\boldsymbol{S} = \boldsymbol{S}^{**}$). Then, the process is repeated until LS is not able to find a neighbouring solution that improves $\boldsymbol{S}$.

3.4 Tabu Search

TS is a metaheuristic that uses memory to guide LS during the algorithm execution. Memory allows TS to escape from local optima and to diversify the search by exploring different regions of the solution space. TS was introduced by Glover (1986) and is explained in detail in Glover (1989, 1990). In addition, TS have been used for the QAP with good results. Among TS for the QAP are the simple TS of Skorin-Kapov (1990), the robust TS (Taillard, 1991), the reactive TS (Battiti and Tecchiolli, 1994) and the TS with mutation operators for solution diversification (Misevicius, 2005).

The main components of the TS heuristic are the tabu list (TLIST), the Tenure Length (TL), the aspiration criterion, the stopping criterion, and the parameter x. TLIST keeps track of the most recent moves. That is, every time a new current solution $\boldsymbol{S}$ is obtained, TLIST is updated accordingly to the move leading to $\mathbf{S}^{**}$. Table 5 shows TLIST after exchanging machines at locations 2 and 3 in $\boldsymbol{S} = (3, 2, 1, 4, 2, 3)$ that leads to $\boldsymbol{S}^{**} = (3, \mathbf{1}, \mathbf{2}, 4, 2, 3)$. Columns in the list represent locations and rows represent machine types. Therefore, according to the list, a machine of type 2 cannot visit location 2 for the next TL iterations, and a machine of type 1 cannot visit location 3 for the next TL iterations. Therefore, TL is the number of iterations a move is tabu. That is, if a machine of certain type left a location, a machine of the same type cannot be assigned to that location for the next TL iterations.

Table 5 Tabu list example

	L1	*L2*	*L3*	*L4*	*L5*	*L6*
M1	–	–	TL	–	–	–
M2	–	TL	–	–	–	–
M3	–	–	–	–	–	–
M4	–	–	–	–	–	–

The aspiration criterion removes a move from TLIST if the move leads to the best solution ever found (i.e. $\boldsymbol{S}^{best}$). The stopping criterion terminates the algorithm after certain number of consecutive iterations (ITER) without improvement (STOPITER). Finally, the parameter x, is used to decide if $f(\boldsymbol{S}^*)$ is obtained by solving the MCMFP optimally, or if $f(\boldsymbol{S}^*)$ is estimated by fixing flows between machines and updating $f(\boldsymbol{S}^*)$ accordingly. The main idea behind x is to solve the MCMFP optimally when the search reaches promising areas of the solution space. On the contrary, when the search arrives to areas with low quality solutions, $f(\boldsymbol{S}^*)$ is estimated using the alternative method explained above. More exactly, if the difference between $f(\boldsymbol{S})$ and $f(\mathbf{S}^{best})$ is less than x, the MCMFP is solved optimally. Otherwise, flows between machines are kept fixed, and $f(\boldsymbol{S}^*)$ is estimated accordingly. Following is an outline of the TS heuristic for the IMALP

Step 1. Initialise heuristic parameters: TLIST, TL, x, ITER, STOPITER.

Step 2. Obtain an initial solution, $\boldsymbol{S}^0$, using one of the construction algorithms given above (used CA1).

Set $\boldsymbol{S}^{best} = \boldsymbol{S}^0$, and set $f(\boldsymbol{S}^{best}) = f(\boldsymbol{S}^0)$.

Set $\boldsymbol{S} = \boldsymbol{S}^0$, and set $f(\boldsymbol{S}) = f(\boldsymbol{S}^0)$.

Step 3. Evaluate all possible pairwise exchanges for solution $\boldsymbol{S}$.

If $f(\boldsymbol{S}) - f(\boldsymbol{S}^{best}) \geq x$

Fix product flows between machines and estimate $f(\boldsymbol{S}^*)$ for each neighbouring solution as explained above.

Else

Solve MCMFP optimally to find $f(\boldsymbol{S}^*)$ for each move.

Select best admissible move (defined as move*), and perform move* on solution $\boldsymbol{S}$ to obtain new solution $\boldsymbol{S}^{**}$. Best admissible move is defined as the best non-tabu move or the tabu move that overrides tabu restriction.

Set $\boldsymbol{S} = \boldsymbol{S}^{**}$, and set $f(\boldsymbol{S}) = f(\boldsymbol{S}^{**})$.

If $f(\boldsymbol{S}^{**}) < f(\boldsymbol{S}^{best})$

Update $f(\boldsymbol{S}^{best}) = f(\boldsymbol{S}^{**})$, and set $\boldsymbol{S}^{best} = \boldsymbol{S}^*$.

Set ITER = 0

Else

ITER = ITER + 1

Step 4. Update TLIST as explained above.

Step 5. If ITER ≤ STOPITER.

Go to step 3.

Else

Terminate the search.

3.5 Memetic Algorithm

Memetic Algorithms (MA) can be defined as evolutionary algorithms combined with one or more LS techniques. (Krasnogor and Smith, 2005). MA was presented first by Norman and Moscato (1989). Moreover, MA combines the diversification strengths of Genetic Algorithms (GA) with the effectiveness of LS techniques. Recently, Drezner (2007) presented a MA for the QAP that combines GA with TS. Drezner (2007) reported having obtained the best results known in 99.4% of a set of difficult problem instances from the QAP literature.

The GA component of MA presented in this research is based on Mendes, Goncalves and Resende (2008), and the LS technique is the same TS heuristic described in Section 3.4. The main components of GA are the population $\boldsymbol{P}$, the evolution mechanism and the stopping criterion. $\boldsymbol{P}$ consists of a group of chromosomes. Each chromosome represents a solution $\boldsymbol{S}_k$, and each solution $\boldsymbol{S}_k$ is composed of genes. Each gene is denoted as $s(i)$ where i represents a location in $\boldsymbol{S}_k$, and $s(i)$ represents the type of machine assigned to location i. An example of a chromosome is $\boldsymbol{S} = (1, 3, 3, 4, 2, 2)$, and $s(5) = 2$ is an example of a gene (i.e. a machine of type 2 is assigned to location 5). Finally, the evolution mechanism is responsible for the evolution of $\boldsymbol{P}$ towards a more fit population.

First, MA starts by generating an initial $\boldsymbol{P} = \{\boldsymbol{S}_1, ..., \boldsymbol{S}_p\}$. Each chromosome in $\boldsymbol{P}$ is generated using CA2. Moreover, each chromosome is improved using TS before entering the population $\boldsymbol{P}$. In other words, each member of $\boldsymbol{P}$ is an improved chromosome. Once an initial $\boldsymbol{P}$ is obtained, the evolution mechanism is applied to $\boldsymbol{P}$ in order to generate a new population $\boldsymbol{P}^*$. The evolution mechanism consists of three components. These components are Elitism, Breeding and Mutation. Elitism passes the best chromosome in $\boldsymbol{P}$ (i.e. $\boldsymbol{S}_k$ with the lowest $f(\boldsymbol{S}_k)$) to $\boldsymbol{P}^*$. Elitism keeps the fittest chromosomes in the population, making their genes available to future offsprings. Breeding generates a predefined number of children (i.e. offspring size). Breeding is a modification of the parameterised uniform crossover strategy introduced by Spears and Dejong (1991). Breeding generates one offspring at a time. Table 6 gives an example of breeding. First, two different chromosomes are selected randomly from $\boldsymbol{P}$ (i.e. $\boldsymbol{S}_r$ and $\boldsymbol{S}_t$). Second, genes from the first parent, $\boldsymbol{S}_r$, are passed to the offspring according to a certain probability (i.e. passprob). More specifically, a random number between 0 and 1 is generated for each gene of parent $\boldsymbol{S}_r$. If the random number < passprob for a gene, then the gene is passed to the offspring. Notice that $s(1) = 1$, $s(2) = 2$, $s(3) = 4$, $s(5) = 2$ are passed to the offspring $\boldsymbol{S}^*$. That is, so far $\boldsymbol{S}^* = (1, 2, 4, \_, 2, \_)$. Third, the remaining genes (i.e. genes that were not inherited by the offspring from $\boldsymbol{S}_r$) are passed from $\boldsymbol{S}_t$ or randomly assigned from the unassigned machines (i.e. $\boldsymbol{M} - \boldsymbol{S}^*$), such that infeasible solutions are not generated, which is often the drawback of GAs. Notice that $s(4)$ from parent 2 cannot be assigned to the offspring, since there is only one machine of type 4 in $\boldsymbol{M} = \{1, 2, 2, 3, 3, 4\}$. On the other hand, the offspring can inherit $s(6) = 3$ from $\boldsymbol{S}_t$. Since, there is one unassigned location (i.e. $s(4)$) and only one unassigned machine in $\boldsymbol{M}$ (i.e. a machine of

type 3), then that machine is assigned to the location (i.e. $s(4) = 3$).The final offspring is $\boldsymbol{S}^* = (1, 2, 4, 3, 2, 3)$. The last step is to improve $\boldsymbol{S}^*$ with TS before adding it to $\boldsymbol{P}^*$.

Table 6 Breeding example

Machine set: M = {1, 2, 2, 3, 3, 4}						
Parent 1 Chromosome $(\boldsymbol{S}_r)$	1	2	4	3	2	3
Parent 2 Chromosome $(\boldsymbol{S}_t)$	2	1	3	4	2	3
Random number (0, 1)	0.67	0.34	0.15	0.86	0.45	0.81
Probability < 0.7 = *passprob*	Yes	Yes	Yes	No	Yes	No
Offspring chromosome S*	1	2	4	3	2	3

Mutation generates a certain number of chromosomes (i.e. Mutated) using CA2. Each one of the Mutated chromosomes is improved using TS before being added to the new population $\boldsymbol{P}^*$. Mutation brings mutated chromosomes that are hopefully not related to the ones in $\boldsymbol{P}$. These mutated chromosomes contribute to keeping the population from converging to a restricted area of the solution space too soon. In addition, the application of TS minimises the impact of small changes in existing chromosomes (i.e. traditional mutation strategies). In fact, changing one or two genes of a chromosome would generate a solution that belongs to the chromosome's neighbourhood. Moreover, since TS is previously applied to the chromosome, the chromosome neighbourhood has already been explored. When a new population $\boldsymbol{P}^*$ is generated, $\boldsymbol{P}^*$ becomes $\boldsymbol{P}$ and the evolution mechanism is applied again until the stopping criterion is reached. The MA stopping criterion is to run the heuristic for a certain amount of computational time (stoptime). Following is an outline of the MA heuristic for the IMALP.

Step 1. Initialise parameters: passprob, ClockTime, StopTime, Elite, OffspringSize, Mutated.

Set PopulationSize = Elite + OffspringSize + Mutated.

Step 2. Generate an initial population $\boldsymbol{P}$.

Generate PopulationSize chromosomes using CA2.

Apply above TS to each chromosome.

Add chromosomes to $\boldsymbol{P}$.

Step 3. Apply evolution mechanism.

Select the best Elite chromosomes (chromosomes with lowest $f(\boldsymbol{S})$).
Add the Elite chromosomes to $\boldsymbol{P}^*$.

Generate OffspringSize Offsprings as follows:

Select $\boldsymbol{S}_r$ and $\boldsymbol{S}_t$ randomly from $\boldsymbol{P}$ such that $\boldsymbol{S}_r \neq \boldsymbol{S}_t$.

Generate $\boldsymbol{S}^*$ using the modified parameterised uniform crossover strategy.

Apply TS to $\boldsymbol{S}^*$.

Add $\boldsymbol{S}^*$ to $\boldsymbol{P}^*$.

Generate mutated chromosomes using CA2.

Apply TS to mutated chromosomes.

Add mutated chromosomes to $\boldsymbol{P}^*$.

Set $\boldsymbol{P} = \boldsymbol{P}^*$.

Step 4. If ClockTime < StopTime.

Go to step 3.

Else

Terminate the algorithm.

4 Computational results

The metaheuristics described above were evaluated with the dataset generated in Urban, Chiang and Rusell (2000). The dataset consists of 21 test problems. These problem instances were designed using layouts from Nugent, Vollman and Ruml (1968) with 6, 8, 9, 12, 15, 20 and 30 locations. In addition, levels of 3, 6 and 9 different products were considered for each layout size.

The algorithms were coded in Visual Basic 2005, and were evaluated in a computer equipped with a 2.2 MHz AMD Turion 64 processor with 1 GB of memory and windows XP. The solver used for the MCMFP linear programme was lp_solve (version 5.5.0.10). lp_solve is an open source (mixed integer) linear programming system which was developed by Berkelaar, Eikland and Notebaert (2007). Finally, optimal solutions for small size problem instances were obtained using the IMALP formulation presented above and the CPLEX solver version 6.6.

TS initial solutions were provided using CA1. TS heuristic parameter settings were obtained through experimentation and they were set as follows. TL was set as 0.05* machine types* locations. ITER was set as machine types* locations. The parameter x was set in such a way that MCMFP was solved optimally for approximately 50% of the visited solutions, for the other 50% $f(\boldsymbol{S}^*)$ was estimated keeping product flows fixed. Similarly, MA parameters were obtained through experimentation and were set as follows. $\boldsymbol{P} = 10$, Elite = 1, OffspringSize = 6, Mutated = 3, probbreed = 0.5. In addition, the parameters of the TS heuristic embedded within the MA were set as before with the exception of ITER, which was set as ITER = locations. MA stopping criterion was set as one-third of the time used by TS. That is, 3 runs of MA used the same computational time as 1 run of TS so that a fair comparison is made between the performances of the proposed heuristics.

Table 7 summarises the results obtained. The first column is the problem identification. The second column corresponds to the best OFV obtained by the GRASP heuristics presented in Urban, Chiang and Rusell (2000). The third column is the computational time used by one run of the TS heuristic and three runs of MA combined. The fourth column gives the results obtained using the proposed TS heuristic and the last 3 columns correspond to the OFV obtained for each one of the three different runs using the MA. Notice that values in bold represent best found solutions. Also, values with asterisk represent OFV of optimal solutions obtained with the IMALP formulation presented above. Comparing TS with GRASP, it can be seen that TS performed better in 9 instances, equal in 10 instances and worse in 1 instance. Similarly, when comparing MA with GRASP, MA obtained same solutions for 10 problems and better solutions in the remaining 11. Also, when comparing MA with TS, it can be seen that MA obtained better results in three instances and same results in the remaining 18 instances. Moreover,

MA obtained the best solutions in all the runs with the exception of the first run of instance 19. This is an indication that the size problems in the dataset is relatively small.

Table 7 Heuristics results

Inst	*GRASP*	*Time*	*TS*		*MA*	
1	**243.33***	0.01	**243.33***	**243.33***	**243.33***	**243.33***
2	**1,317.2***	0.01	**1,317.20***	**1,317.20***	**1,317.20***	**1,317.20***
3	**320.00***	0.01	**320.00***	**320.00***	**320.00***	**320.00***
4	**357.50***	0.01	**357.50***	**357.50***	**357.50***	**357.50***
5	**6,708.33***	0.01	**6,708.33***	**6,708.33***	**6,708.33***	**6,708.33***
6	**558.00***	0.02	**558.00***	**558.00***	**558.00***	**558.00***
7	241	0.02	**239.29***	**239.29***	**239.29***	**239.29***
8	110.67	0.03	**107.45***	**107.45***	**107.45***	**107.45***
9	**3,298.00***	0.03	**3,298.00***	**3,298.00***	**3,298.00***	**3,298.00***
10	**55.09***	0.06	**55.09***	**55.09***	**55.09***	**55.09***
11	1,008.00	0.12	**966**	**966**	**966**	**966**
12	**6,051.38**	0.34	**6,051.38**	**6,051.38**	**6,051.38**	**6,051.38**
13	**5,590.00**	0.17	**5,590.00**	**5,590.00**	**5,590.00**	**5,590.00**
14	2,193.63	0.59	2,186.96	**2,184.96**	**2,184.96**	**2,184.96**
15	623.08	0.95	619.11	**613.1**	**613.1**	**613.1**
16	1,245.80	0.9	**1,232.28**	**1,232.28**	**1,232.28**	**1,232.28**
17	425.7	3.18	**422.32**	**422.32**	**422.32**	**422.32**
18	13,100.00	3.24	**13,100.00**	**13,100.00**	**13,100.00**	**13,100.00**
19	13,157.98	14.63	**12,871.08**	12,900.76	**12,871.08**	**12,871.08**
20	2,111.00	32.78	2,163.25	**2055**	**2055**	**2055**
21	1,373.52	103.58	**1,350.31**	**1,350.31**	**1,350.31**	**1,350.31**

Finally, notice that TS and MA improved the best-known results in 52.4% of the instances and equaled the best-known results in the remaining 47.6% of the cases. These results suggest that solving the IMALP iteratively (i.e. solving MLP while keeping flows between machines fixed) restricts the search leaving promising areas of the solution space unexplored. Also it is important to mention that solving the MCMFP for all machine layouts is computationally expensive, leading to impractical algorithms. Therefore, the parameter x added to TS has an important role in the speed of the algorithm. Recall, if the cost of a solution is within x of the best found solution, the actual costs of the neighbouring solutions are obtained. Otherwise, the costs of the neighbouring solutions are estimated. More generally, the heuristic allows exploring promising areas of the solution space with detail and to leave non-promising areas of the solution space more quickly. In Addition, the strong performance of MA is explained by the combination of the diversification strengths of GA with the search ability of TS. That is, GA provides many different initial solutions (i.e. machine layouts) and TS search the neighbourhood of each one of these solutions effectively. Notice that each new solution (i.e. offspring) provided by GA represents a promising area of the solution space, since it is generated by

combining two good solutions (i.e. well fit parents). Also, since GA generates one mutated solution at each iteration, MA is prevented from converging to local optima to soon.

5 Conclusions

In this article, a mathematical formulation, a TS and a MA heuristics are presented for the IMALP. Both heuristics showed superior performance than the GRASP heuristic presented in Urban, Chiang and Rusell (2000). Also the MA algorithm performed better than the TS heuristic. For future research, it is recommended to develop a more extensive data set that considers layouts with a larger number of locations and product routes that resemble job shops and machine/part families. In addition, relax some of the problem assumptions such as determine the number of machine types and replicas instead of the number and type of machines given as an input. Also, maximise profits such that demand is not met.

Acknowledgements

The authors express sincere thanks to Dr. Timothy Urban for providing the IMALP dataset. Also, the authors especially thank the anonymous reviewers for their valuable suggestions.

References

Battiti, T. and Tecchiolli, G. (1994) 'The reactive tabu search', *ORSA Journal of Computing*, Vol. 6, pp.126–140.

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

Castillo, I. and Peters, B. (2003) 'An extended distance based facility layout problem', *Int. J. Production Research*, Vol. 41, pp.2451–2479.

Drezner, Z. (2007) 'Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem', *Computers and Operations Research*, Vol. 35, pp.717–736.

Foulds, L.R. (2005) 'Dynamic network flow models of sustainable grain silo locations', *Int. J. Operational Research*, Vol. 1, pp.74–88.

Glover, F. (1986) 'Future paths for integer programming and links to artificial intelligence', *Computers and Operations Research*, Vol. 13, pp.533–549.

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

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

Jaramillo, J.R. and McKendall, A.R. (2004) 'Dynamic extended facility layout problem', *IRC Annual Conference*, Houston, TX.

Kennington, J.L. (1978) 'A survey of linear cost multicommodity network flows', *Operations Research*, Vol. 26, pp.209–236.

Koopmans, T.C. and Beckmann, M.J. (1957) 'Assignment problems and the location on economic activities', *Econometrica*, Vol. 25, pp.53–76.

Krasnogor, N. and Smith, J.E. (2005) 'A tutorial for competent memetic algorithms: model, taxonomy and design issues', *IEEE transactions on Evolutionary Computation*, Vol. 9, pp.474–488.

Mendes, J.J., Goncalves, J.F. and Resende, M.G. (2008) 'A genetic algorithm for the resource constrained multi-project scheduling problems', *European Journal of Operational Research*, Vol. 189, pp.1171–1190.

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

Norman, M.G. and Moscato, P. (1989) 'A competitive and cooperative approach to complex combinatorial search', *Caltech Concurrent Computation Program: Report 826*. Pasadena, CA.

Nugent, C.E., Vollman, T.E. and Ruml, J. (1968) 'An experimental comparison of techniques for the assignment of facilities to locations', *Operations Research*, Vol. 16, pp.150–173.

Robacker, J.T. (1955) 'On network theory', *The RAND Corporation: Report RM 1498*. Santa Monica, CA.

Sahni, S. and Gonzales, T. (1976) 'P complete approximation problems', *Journal of the Association of Computing Machinery*, Vol. 23, pp.555–565.

Singh, S.S. and Sharma, R.R. (2006) 'A review of different approaches to the facility layout problems', *Int. J. Advanced Manufacturing*, Vol. 30, pp.425–433.

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

Spears, W.M. and Dejong, K.A. (1991) 'On the virtues of parameterised uniform crossover', Paper presented in the Proceedings of the *4th International Conference on Genetic Algorithms*, pp.230–236.

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

Urban, T.L., Chiang, W-C. and Rusell, T. (2000) 'The integrated machine allocation and layout problem', *Int. J. Production Research*, Vol. 38, pp.2911–2930.