Int J Adv Manuf Technol (2001) 17:593–602

The International Journal of
Advanced
Manufacturing
Technology

Simulated Annealing Combined with a Constructive Algorithm for Optimising Assembly Workcell Layout

D. Barral[1,2], J.-P. Perrin[1], E. Dombre[2] and A. Liégeois[2]

[1]Dassault Systemes, 9, quai Marcel Dassault, 92150 Suresnes, France; and [2]LIRMM, UMR 5506, CNRS/Université Montpellier II, 161 rue Ada, 34090 Montpellier, France

This paper addresses the development of a tool for optimising assembly workcell layout in the context of an industrial robotic CAD/CAM/CAE software product. The criterion to be minimised is the cycle time for completing a given sequence of operations, which is achieved by determining the relative positions of peripheral machines on the cell floor. The algorithm is constructive: the machines are placed one at a time in the robot neighbourhood, by means of a modified simulated annealing (SA) method. This method yields several possible and optimal positions for a machine, and several layouts are thus obtained at the end of execution. The optimisation tool has been implemented in IGRIP, and a case study illustrates its performance.

Keywords: Assembly workcell layout; CAD/CAM/CAE; Constructive approach; Simulated annealing

1. Introduction

In assembly task applications, robotic manipulators perform a repetitive sequence of operations on and between a set of peripheral machines. The productivity of such tasks can be considerably improved by minimising the cycle time.

For a given manipulator, cycle time depends on many parameters, such as the position of the manipulator relative to the task, the sequence in which the operations are performed, the maximum velocities and accelerations of the actuators, or the configuration of the arm along the path. It also depends greatly on the relative position of the points that the robot must access, and, thus, on the workcell layout in the case of an assembly task: the robot must stop at the access points of different machines with which it interacts.

The design process used so far in industry for robot workcell layout contains some interactive check and change loops that must be performed by the user. Such a process is time consuming and the quality of the layout depends heavily on the human designer [1,2]. To achieve an integrated design process for workcells, an automatic layout-planning tool is required in CAD/CAM/CAE systems.

Although robotic assembly workcells are widespread in industry, the problem of optimising their layout has received surprisingly little attention. Chittajallu and Sommer [3] first locate odd-shaped devices circumscribed in circles to minimise a statistically weighted distance function. Then, the relative orientations of the devices are calculated. Although the circle representation provides a simple means of detecting overlapping of machines, there is much excess space circumscribed by the circle in practical cases, and the devices need to be brought closer to one another and reoriented. Lueth [4] proposes an approach to plan robot workcells in the Cartesian configuration space, where the problem is formulated as the quadratic set covering problem (QSP) [5]. An algorithm minimises the length of all the collision-free paths required, and the machines are placed one at a time using a local optimisation algorithm. Tay and Ngoi [6] describe a heuristic algorithm that attempts to minimise the total Euclidean distance travelled by a robot arm to perform a given sequence of operations. The method applied is constructive: the machines are positioned and oriented one at a time, according to a limited list of possible empty areas. The robot type that can be considered is limited to Cartesian structures. Finally, Drezner and Nof [7] study the problem of planning a robot assembly station in which component parts are picked from bins and assembled. Several approaches to the problem of optimising the bin organisation, the picking sequence and the route are considered. Nevertheless, they do not attempt to optimise the layout of every component in the cell, and the problems considered are rather different.

The well-known facility layout problem (FLP) is very similar to the robotic assembly workcell layout problem, and the latter can be considered as an extension in the 3D world of the former. A standard form of FLP involves n facilities (indexed $1,\ldots,n$), an area a_i for each facility i, and a fixed flow f_{ij} between each distinct pair of facilities. The problem is known to be NP-hard [8], and optimisation methods can be successful only in cases of simple cells. Early work on the FLP used the rather inflexible "cell assignment" representation strategy and focused on constructive algorithms and/or iterative improve-

Correspondence and offprint requests to: D. Barral, SYNOPSYS, Inc., 1025 rue H. Becquerel, 34000 Montpellier, France.

ment methods [5,9–11], where an "iterative improvement method" consists in improving an initial layout by successive pairwise exchanges of facilities.

FLP research has since progressed by using a more sophisticated problem solving framework, such as genetic algorithms (GAs) [12–14] or simulated annealing (SA) [15–18], and by using better layout representations [18].

In this paper, we present a constructive algorithm to optimise the layout of a robotic assembly workcell that consists of a robot and peripheral machines. As mentioned above, the problem is an extension in the 3D world of FLP, and it can thus be considered at least as complex. The criterion considered is strictly the cycle time of a task. The algorithm uses a modified SA method [19] that yields several possible placements for a machine, and several layouts are thus obtained. All the constraints (accessibility of the machine access points, collision avoidance, and feasibility of the task) are handled, in order to produce reliable solutions to real cases in the context of an industrial CAD/CAM/CAE system.

2. The Assembly Workcell Layout Problem

The assembly workcell layout problem consists in determining the relative positions of peripheral machines in the neighbourhood of the robot that interacts with them. The aim is to optimise one or more of the workcell performance indices. In this paper, we consider a single criterion: the cycle time corresponding to the assembly task that has to be fulfilled. The effort required for calculating cycle time can be quite substantial in an industrial CAD/CAM/CAE system. Indeed, each evaluation implies a simulation of the robot movement, as controller manufacturers provide their own simulation modules for their robots.

The following factors must be considered [6]:

1. The shape of the machines can be complicated. This introduces the problem of finding a method of representation that limits the number of possible orientations of the machines relative to the robot, and raises the problem of detecting the overlapping of the machines efficienlty.
2. Every machine has at least one access/delivery point, which is the point that a robot must reach to carry out the task associated with the machine.
3. Machines should be arranged with enough space between them to allow for maintenance, repair, or operator movement. The clearances may differ for different machines.
4. There may be more than one robot involved in the assembly task.
5. The machine access/delivery points must be placed in the workspace of the robot that must reach them.

The general assembly workcell layout problem is very complex and combinatorial (FLP is NP-hard). Therefore, some simplifications and assumptions are made here which do not prevent the resolution of most practical cases [6]:

1. The workcell consists of one fixed-based robot and a set of peripheral machines.
2. The operation sequence associated with the assembly task is fixed.
3. The type and shape of machine considered are not limited.
4. A machine has only one access/delivery point. This assumption is reasonable as most machines have only one such point. The position of the access point on a machine is not limited at the "centre" of the machine.
5. Space clearance between machines is to be specified by the user. The internal representation of a machine will be determined taking account of the space allowances.
6. The floor space available for computing the layout is assumed to be unlimited. Slight shifting of the machines to suit the available floor space will not affect the result significantly. Moreover, the layout may be displaced as a whole to fit a given floor area.

To provide a reliable solution to practical and industrial cases, the following constraints must be taken into account:

1. Accessibility of the machine access points.
2. Collision avoidance between the robot and its environment, when the robot tool reaches the point where work is to be performed.
3. Collision avoidance between the robot and its environment, when the robot arm is moving from access point to access point.

Finally, the following initial data are to be provided by the user: the assembly robot model, the geometry of the machines with which it interacts, and the operation/interaction sequence corresponding to the assembly task to be performed.

3. Definition of an Operation Sequence

Noting the inadequacy of standard CAD/CAM/CAE tools to describe a sequence of operations associated with an assembly task, a new user interface was designed to minimise the number of interactions necessary to the user.

This interface allows the user to define what are termed "machine–machine interactions", which are to be performed by the robot, instead of work points in the robot workspace. Such interactions occur when the robot brings an element (a piece or a tool) from a given machine to a second one. An interaction where the robot moves between machines without carrying anything is not a machine–machine interaction. Indeed, such an interaction can be deduced from the previous ones, and it is not necessary to specify it explicitly.

Let us now consider a simple example. The workcell consists of a robot and three peripheral machines (machine 1, machine 2, and machine 3). The task to be performed is as follows: grasp a piece from machine 1 twice in succession; place the pieces on machine 2; grasp a tool on machine 3 and assemble the pieces on machine 2; bring the tool back to machine 3.

The interface allows the user to describe a machine–machine interaction by simply clicking successively on the initial and goal machine names, and by entering the number of repetitions of the interaction. These steps are illustrated in Fig. 1 for the interaction between machines 1 and 2.

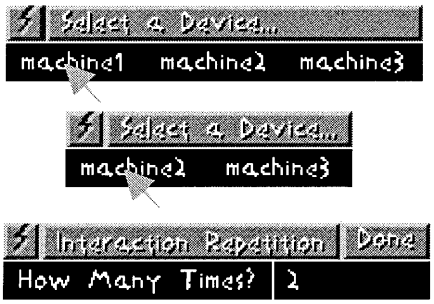

Fig. 1. Machine–machine interaction definition.

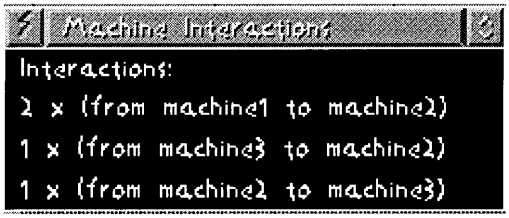

Fig. 2. Interaction sequence description.

While describing the interaction sequence, the user can check the interactions defined so far, as shown in Fig. 2. All the interactions are displayed.

The machine sequence (i.e. the order in which the robot must go to the machines) is then deduced directly from the interaction sequence. For the example used so far, the sequence is the following:

mac.1-mac.2-mac.1-mac.2-mac.3-mac.2-mac.3

Next, a rank table, that is used to calculate the cycle time, is automatically generated (Fig. 3). The *i,i* element of the table is the number of times the robot accesses machine *i*. The *i,j* and *j,i* elements correspond to the number of movements (not interactions) between machine *i* and machine *j*, whatever the order. For instance, the 1,2 element is equal to 3: the robot goes first from machine 1 to machine 2, then from machine 2 to machine 1, and finally from machine 1 to machine 2 again.

4. Workcell Internal Representation

4.1 Machine Geometry Simplification

The 2D machine internal representation proposed is aimed at simplifying the numerous collision calculations during optimisation. It describes exactly the shape of all rectilinear machines, i.e. machines composed of some low-level rectangular elements (rectangles, for short), such as rectangle-shaped, L-shaped or T-shaped machines. Shapes such as circles or polygons are represented only by bounding rectangles, though, such an approximation is minimal in that the shape of most machines is rectilinear.

machine n°	1	2	3
1	**2**	3	0
2	3	**3**	3
3	0	3	**2**

Fig. 3. The rank table.

First, the 2D approximate contours of the machines on the floor space are calculated:

$$B = \{b_i / 1 \le i \le m\}$$

where
b_i is the bounding rectangle of machine i
m is the number of machines

Next, the set of rectangles that makes up b_i is deduced, if necessary, from the 3D initial representation of the machine i:

$$B_i = \{b_{ij} / 1 \le j \le m_i\}$$

where
b_{ij} is the decomposed rectangle j for the machine i
m_i is the number of decomposed rectangles for the machine i

For each partitioned rectangle b_{ij}, x_{ij}, and y_{ij} are the x- and y-coordinates of the centre of rectangle b_{ij}. Let w_{ij} and l_{ij} be the width and the length of rectangle b_{ij}. Unless machine rotation is considered, the structures of their decomposed rectangles would never be changed. We can define the decomposed rectangles in a rectilinear machine as *solid-connected*, in which, the offsets among these rectangles are kept constant. Thus, during the placement process, we should keep the offsets among all the rectangles b_{ij} in a set B_i. For simplification, let xo_{ij} and yo_{ij} be two constants to represent the x and y offsets from the centre of b_i to the centre of b_{ij}. For the bounding rectangle b_i of the machine i, x_i and y_i are the x- and the y-coordinates of the centre of b_i. The width and the length of b_i are w_i and l_i. Then, we can define the centre of a rectilinear machine i as the centre of its related bounding rectangle b_i, and define the x-offset and the y-offset from b_i to b_{ij} as $xo_{ij} = xb_{ij} - x_i$ and $yo_{ij} = yb_{ij} - y_i$, respectively. Figure 4 presents a simple example to demonstrate the relationship between a rectilinear machine and its partitioned rectangular cells. All the variables and constants used in this example are labelled to illustrate the relationship of these notations.

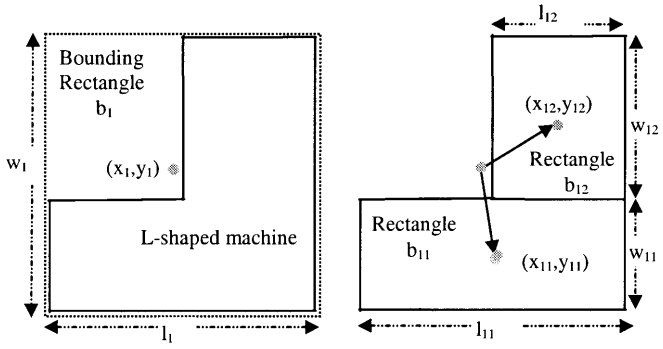

Fig. 4. The relationship between the bounding rectangle b_1 of the rectilinear machine 1, and its corresponding rectangles b_{11} and b_{12}.

4.2 Detection of Overlapping Machines

The previous representation provides a simple function to detect in 2D the overlap of machines. Indeed, the robot workcell layout problem is to minimise the cycle time necessary to achieve a predefined task, under the constraint that the machine overlap index C_{mo} is zero. We can define C_{mo} as follows [20]:

$$C_{mo} = \sum_{i=1}^{m} \sum_{j=1}^{m_i} \sum_{k=i+1}^{m} \sum_{l=1}^{m_k} od_{ij,kl} \tag{1}$$

In (1), the overlap distance, $od_{ij,kl}$, is a symbolic representation of the machine overlap between b_{kl} and b_{ij}. It can be defined as follows:

$$od_{ij,kl} = (id_{ij,kl} - d_{ij,kl}) \times f_h\,(id_{ij,kl} - d_{ij,kl}) \tag{2}$$

$$id_{ij,kl} = \min\left(\frac{lb_{ij} + lb_{kl}}{2 \times |dx_{ij,kl}|}, \frac{wb_{ij} + wb_{kl}}{2 \times |dy_{ij,kl}|}\right) \times d_{ij,kl} \tag{3}$$

where $id_{ij,kl}$ is the ideal minimum distance between b_{kl} and b_{ij} that sets the machine overlap to zero, and function $f_h(.)$ used in (2) is the 0–1-hardlimit threshold function where $f_h(x) = 1$ if $x > 0$, $f_h(x) = 0$ otherwise.

The Euclidean distance $d_{ij,kl}$ between b_{kl} and b_{ij} is defined as:

$$d_{ij,kl} = \sqrt{(dx_{ij,kl}^2 + dy_{ij,kl}^2)} \tag{4}$$

where $dx_{ij,kl} = xb_{kl} - xb_{ij}$ and $dy_{ij,kl} = yb_{kl} - yb_{ij}$ are defined as the x- and y- coordinate displacements from the centre of b_{kl} to the centre of b_{ij}, respectively. Figure 5 presents an example with a simple L-shaped machine and a rectangular machine to demonstrate this representation method.

4.3 Workcell Floor Representation

The workcell layout optimisation method presented in this paper requires the calculation of an initial placement for each machine. This placement must, in particular, avoid machine overlaps.

To determine such placements, a workcell floor representation, derived from the "spatial representation method" presented in [21], is used. A 2D matrix is updated each time a machine is added, representing the current layout of the machines on the floor space. The first row and first column

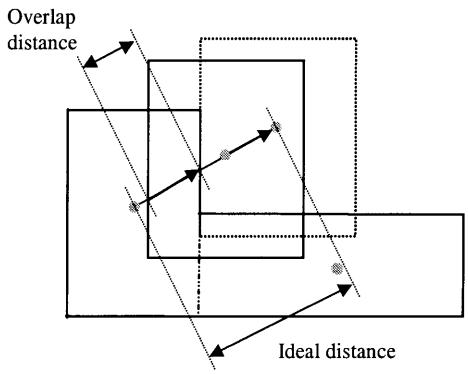

Fig. 5. An illustration of the overlap distance and the ideal distance.

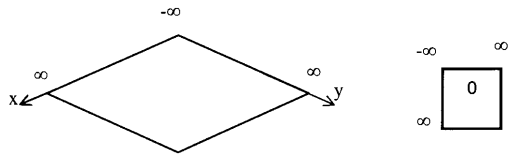

Fig. 6. Empty floor space representation.

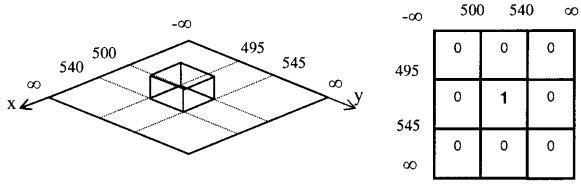

Fig. 7. Representation of a machine placed on the floor space.

of the matrix correspond to the x- and y-coordinates of the layout. Remaining matrix cells contain "*0*" or "*1*", depending on their occupation by a machine ("*0*" represents empty spaces).

Figure 6 shows how the initial floor space is represented. The initial 2D matrix is shown in a schematic manner, for better understanding. As mentioned above, the space available is supposed to be unlimited.

Assume now that a rectangle-shaped machine, named "machine 1", dimensions of which are $l_1 = 40$ and $w_1 = 50$, is placed on the floor such that $x_1 = 520$ and $y_1 = 520$. The 2D matrix expands to a 3×3 matrix, and the floor space is divided into 9 sections indicated by the dotted lines on the floor area (Fig. 7).

If an L-shaped machine, named "machine 2", dimensions of which are $l_{21} = 30$, $w_{21} = 60$, $l_{22} = 40$ and $w_{22} = 30$, is placed such that $x_2 = 300$ and $y_2 = 700$, the matrix obtained is 7×7, as illustrated in Fig. 8.

As more peripheral machines are placed on the floor space, the 2D matrix expands step by step to accommodate the additional information. When a machine is to be placed on the floor, only empty cells are selected. Thus, this floor representation will give a fast way of determining possible initial positions for a machine when the optimisation method is applied.

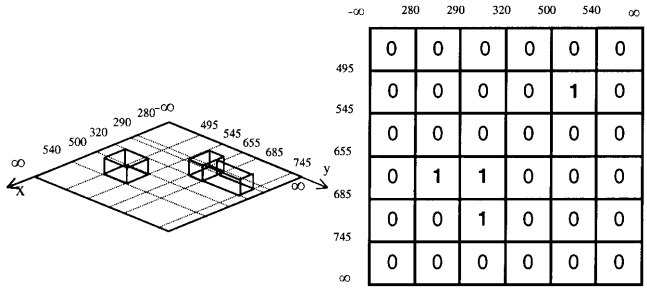

Fig. 8. Representation of two machines placed on the floor space.

5. A Constructive Approach Using a Modified Simulated Annealing Method

5.1 Algorithm Objectives

Whatever the problem to be solved, the basic concept of constructive algorithms consists in setting up step by step, subsets of variables until a complete solution is obtained. This paper concentrates on such a constructive algorithm, where, at each step, a machine is first selected and then located.

A major advantage of these algorithms is the considerably lower computational effort required to execute them, in comparison with iterative improvement methods. Nevertheless, constructive algorithms are, by their nature, not necessarily optimal.

Consequently, the scheme for locating the selected machine must be as unrestrictive as possible and a sufficient number of possible placements must be encountered. Indeed, a bad position for the first machines could lead to a high-cost final workcell layout.

Furthermore, numerous non-reversible choices are made during construction, with no knowledge of their consequences.

Accordingly, the main objectives of the method proposed are:

1. To apply at each step an algorithm able to yield the optimal placement of a machine.
2. To allow multiple possible solutions for the positioning of a machine, and to delay the choice of a solution as much as possible in the optimisation process.

The standard constructive algorithms we propose for optimising assembly workcell layout consists mainly of a rule for selecting a yet unplaced peripheral machine and a scheme for positioning and orienting the selected machine on the floor space. These elements were thus defined with the previous objectives in mind.

5.2 Machine Selection Rule

An existing rule [6] is used to select successively the machines to be placed in the workcell. It proved to be quite efficient, and there was no need to define a new one. We recall its principle in this section.

The selection of the first machine to be placed is rather different from the selection of subsequent machines. It is based on the demand of the machine, i.e. the number of times the robot must access the machine. The reason for using the highest-demand value as the criterion is because a machine with the greatest demand will have the highest interaction with other machines. Thus, it should be placed first, so that other machines that have interactions with it can be placed around it. Should there be more than one machine having the same highest-demand value, the machine with the smallest base area will be chosen.

The subsequent machines are chosen according to the following criterion:

$$C_i = \sum_{j=1}^{n} \frac{I_{ij}}{I_{\max j}} + kS_i \tag{5}$$

where
I_{ij} is the number of interactions of machine i with the already placed machine j ($i \neq j$)
$I_{\max j}$ is the largest interaction value associated with machine j
n is the total number of machines
k is a weighting value assigned to size, and is equal to 0.5
S_i is the normalised base area of machine i, defined as follows:

$$S_i = \frac{A_{\min}}{A_i} \tag{6}$$

where
$A_{\min}$ is the smallest base area among the machines
A_i is the area of the machine i

Note: Equations (5) and (6) are privilege machines with small areas. Hence, large machines will rather be placed at the periphery of the workcell.

5.3 Machine Placement Scheme

The first machine is placed so that the robot is positioned at the centre of an "acceptable base location domain section" [19], as shown in Fig. 9. This choice is purely heuristic: no cost function calculation is possible yet, as there are no machines placed on the floor space.

In [6], the method proposed for placing the other machines yields a single solution. Furthermore, this solution is selected after exhaustive evaluation of a search space, limited arbitrarily to make machine boundaries coincide. Indeed, as the criterion to optimise is the cycle time, nothing ensures that machine boundaries coincide. The placement obtained is certainly the optimal solution in the restricted search space, but one can wonder why it would be at least locally optimal in the space of all the possible positions for a machine.

Consequently, we propose to use a modified simulated annealing algorithm, initially designed for optimising welding robot placement [19]. This algorithm will be described below, following a brief review of the standard SA method.

The concept of the SA method was described for the first time in [22]. It involves the Metropolis rule [23] and occurs in the following way. First, a starting solution for the optimisation is chosen, and labelled as the current solution. Next, a new solution is picked randomly in the neighbourhood of the solution. If the new solution has a lower function value than the current point, it is automatically adopted as the "current" solution for the next step. If not, then a random number is

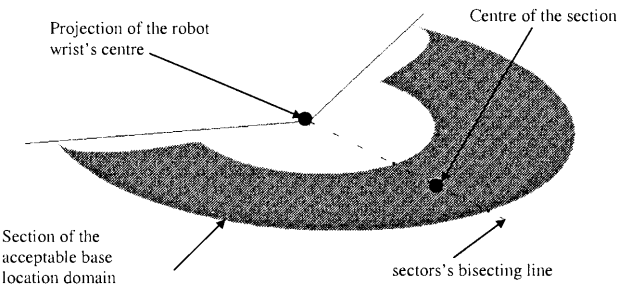

Fig. 9. "Centre" of an "acceptable base location domain section".

computed. That random number determines whether the new solution will be adopted as the current solution. This gives SA the ability to jump out of local minima.

The implementation of SA involves prescribing three parameters: the probability that a new solution will be accepted; the so-called temperature reduction function, and the number of temperature reductions. They are described as follows:

1. *The probability that a new solution will be accepted*. If the difference between the cost of the new solution j and the cost of the current solution i is less than zero, then the probability of acceptance is 1. If the difference is greater than zero, the probability of acceptance is

$$p(\Delta c_{ij},T) = \exp\left(\frac{-\Delta c_{ij}}{T}\right) \tag{7}$$

where

$$\Delta c_{ij} = c(j) - c(i)$$

T is simply a control parameter, which is refered to as the "temperature", in the same units as the cost function.

In the beginning, the value of T is relatively large, so that many cost-increasing moves are accepted. During the optimisation process the temperature is decreased gradually so that fewer and fewer costly moves are accepted.

2. *The temperature reduction function*. Kirkpatrick [22] proposed a rate of temperature reduction of 0.95. Sechen [24] pointed out that the system required fewer state changes at high or low temperatures. However, state change is crucial at a medium temperature. Therefore, the temperature reduction rate can be set at 0.8 at the highest and lowest temperatures, but at 0.95 at a medium temperature. Thus,

$$T_n = \alpha(T)T_{n-1} \quad (0 < \alpha(T) > 1) \tag{8}$$

where $\alpha(T)$ denotes the temperature reduction rate.

The initial temperature is empirically chosen so as to accept the first ten layouts encountered at the beginning of the calculation.

3. *The number of temperature reductions*. The aim is to decrease the temperature to 5% of its initial value. Therefore, the number of temperature reductions can be deduced directly from the temperature reduction function (8).

As mentioned previously, the algorithm used involves changes and enhancements to the SA method, so as to improve the method by increasing the probability of finding the optimal placement for a peripheral machine. These changes were made for several reasons. Obtaining several possible positions for the selected machine fulfils one of the initial objectives: i.e. multiple choices for the positioning of a machine are made possible.

In addition, SA requires long computation times. One reason for the extensive computation is that SA cannot distinguish a local minimum from an ordinary solution, unless it visits the very bottom of the local minimum. During the initial phase, when the temperature is high, SA samples a large area of the search space, but goes deep into the local only much later, when the temperature has been lowered. This means that, even if the machine enters the global minimum's attraction valley early, SA will not perform a local search, but will continue to search the entire domain. This is highly undesirable. It would be advantageous to build some "opportunism" into the conventional SA method, so that it commits itself to local explorations in the early stages of the search.

The optimisation algorithm was designed to achieve this purpose, with two new features:

1. *Freeze-heat cycles*. In order to produce a solution to the local-exploration problem, two factors are important. The first is that, committing to a local search, means, in practice, performing a gradient descent. The second factor is that SA is equivalent to a gradient descent when the temperature is close to zero. Thus, in order to add opportunism to the SA algorithm, it is enough to take the temperature down to nearly zero for as long as it is required, to reach the bottom of the local minimum. Afterwards, the temperature can resume its initial decay schedule. This scheme is called the "freeze–heat cycle".
2. *Remembering local minima*. Ideally, it would be advantageous to explore a minimum pit as fast as possible (freezing), escape from it as fast as possible (heating), and never visit it again. In order to avoid revisiting the same local minima, a list of all the local minima found so far is stored, together with an estimate of their attraction domains. For this purpose, the attraction domain of a local minimum is defined as the set of all the machine locations from which a gradient descent is supposed to terminate at this local minimum. Practically, such sets are impossible to describe analytically, and an attraction domain will correspond to a circle centred on the local minimum, which passes via the initial base location of the gradient descent. Any time later during the search that gradient descent from another location, which is outside an attraction domain A, terminates in a configuration which is inside A, the size of attraction domain A is updated to include the location, resulting in an expanded attraction domain. Such attraction domains may overestimate the size of a real local minimum attraction domain. However, the modified method is formulated so that this does not affect the convergence properties of the SA method, while giving better results. Gradient descent will indeed be initiated only from configurations that are outside any existing attraction domain; while a configuration is within an attraction domain, SA will keep performing random steps, yielding possibly a new minimum.

We recall that the major feature of the modified SA proposed is its ability to find a set of possible placements for the selected machine. Furthermore, the positions calculated are local minima of the cost function, in a search space limited only by the constraints mentioned in Section 2.

The cost function is calculated from Eq. (9) below:

$$F_i = \sum_{j=1}^{n} t_{ij} \times I_{ij} \tag{9}$$

where

t_{ij} is the time necessary for the robot to move from machine i to machine j

I_{ij} is the number of interactions of machine i with the already placed machine j ($i \neq j$)

n is the number of already placed machines
Note: the value of t_{ij} is obtained using the standard functionality available in CAD/CAM/CAE systems.

Before evaluating F_i for a given placement of the selected machine, the accessibility of the access point and the absence of overlapping are verified. The feasibility of the robot movements between the placed machines is also guaranteed; the standard functionality used to calculate move times returns an error message if a robot movement is impossible.

The initial machine position and orientation that are used as the initial solution by the modified SA are determined using the workcell floor representation described in Section 4.3. A sufficient number of empty matrix cells are selected, so that one of the boundaries of the machine to be placed coincides with an already placed machine. Even if such a location does not guarantee a good function value, it is privileged at the beginning to minimise the spaces between the machines.

5.4 Partial Layout Selection Rule

At each step of the constructive algorithm, a machine is selected (Section 5.2) and the modified SA is used to determine several possible placements of the selected machine (Section 5.3). Hence, a partially constructed layout, that contains n already placed machines, yields a set of m layouts containing $n + 1$ machines, where m corresponds to the number of possible locations for the machine to place.

Constructing the layouts for all the machine placements found would be computationally explosive. Therefore, a partial layout selection rule is necessary. This rule is an additional feature in comparison with standard constructive methods.

The scheme is as follows: among all the partial layouts generated after a construction step, keep the k best ones. The most efficient value for k proved experimentally to be 5. It yielded the best "cycle time/CPU time" ratio.

At the end of execution, five optimal workcell layouts are thus provided to the user. Such a result is useful since, generally, the geometric modelling of the workcell does not represent the real-world with enough accuracy. Computing a single solution could give a result which is not achievable in the real-world if the neighbourhood of the solution is not well conditioned.

6. A Case Study

6.1 Assembly Workcell Description

Three-dimensional assembly workcell elements (Fig. 10) were designed in IGRIP from the 2D representation of a representative robotic assembly cell proposed in [24] (Fig. 11). This workcell has already been relaid out by Tay and Ngoi [6], considering a Cartesian robot and the Euclidean distance between the machine access points as the criterion to be minimised.

A modification of the example from [25] was made so that it is more realistic: i.e. replacing the SCARA robot used in the example by an anthropomorphic robot (ACMA X58), which is the most widespread robot type in industry.

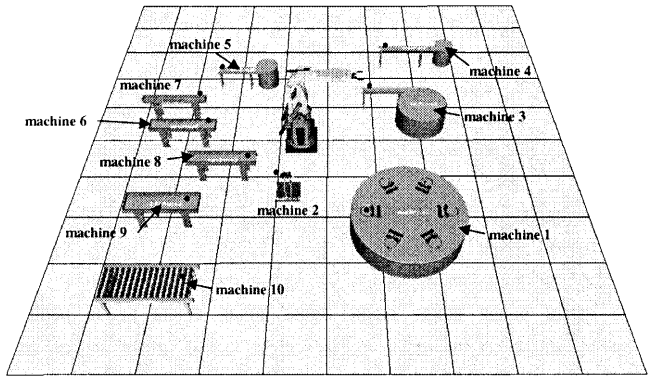

Fig. 10. Three-dimensional assembly workcell components.

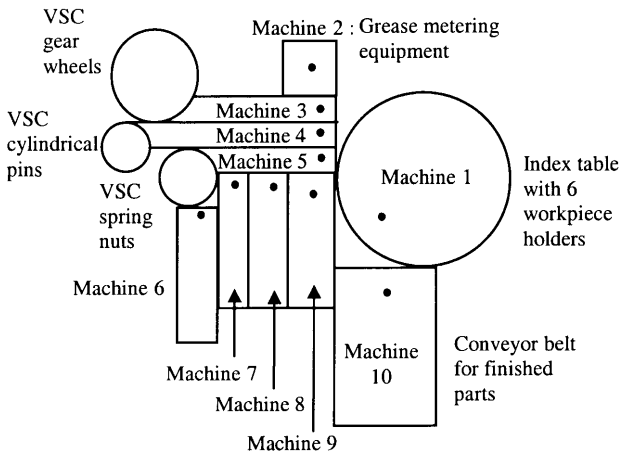

Fig. 11. Initial workcell layout representation in 2D.

The operation sequence of the example is as follows [6]:

1. The robot grasps drive shafts 6 times in succession from machine 6 and places them in the 6 assembly fixtures on the circular index table (machine 1).
2. A combination of bearings and thrust washers is obtained from machine 7 and fitted onto the fixture at machine 1, 6 times in succession.
3. The robot then grasps the grease-metering unit, machine 2, and applies grease to all the housing inner teeth of the 6 fixtures at machine 1.
4. A stepped shaft is grasped 6 times at machine 8 and fitted to the housing at machine 1.
5. Three cylindrical pins are grasped in sequence at machine 4 and fitted into the stepped shaft at machine 1. This procedure is repeated 6 times in sequence.
6. Three gear wheels are grasped 6 times in succession from machine 3 and placed onto the pins.
7. A fan wheel is gripped 6 times at machine 9 and fitted to the drive shaft at machine 1.
8. A spring nut obtained from machine 5 is fitted to each drive shaft at machine 1.

Machine n°	1	2	3	4	5
Size	129600	6400	37050	6400	7550
Normalised size	.035	.719	.124	.719	.609
Machine n°	6	7	8	9	10
Size	11500	4600	11500	16100	29700
Normalised size	.400	1.000	.400	.286	.154

Fig. 12. The size table.

9. The final assembled unit is removed from the index table and placed on the conveyor belt (machine 10).

Figures 12 and 13 show the size and the rank tables generated from the above operation sequence, respectively.

From the previous tables, the values of C_i (see Eq. (5)) can be calculated at each step of the program. The order in which the machines are placed is as follows:

mac.1– mac.7– mac.4– mac.5– mac.9– mac.6– mac.8– mac.3– mac.10–mac.2

6.2 Analysis of the Results

Modified SA is a stochastic method. Hence, results from the algorithm proposed may vary, depending on the execution. Both layouts presented in this section were obtained from a single application, in order to illustrate the results. Figure 14 shows the optimal layout, whereas Figure 15 presents another of the five solutions calculated, which is nearly optimal. Note that there are spaces between the machines for both layouts.

The method proposed by Tay and Ngoi [6] was modified and implemented in IGRIP so that comparison was possible: the robot used is an ACMA X58, instead of a Cartesian robot; the cost function is the cycle time. The layout obtained, which was termed "T&N layout", is shown in Fig. 16.

Table 1 gives the cycle times for the layouts shown in Figs 14, 15, and 16. Table 2 indicates the computation times necessary for both methods for optimising the workcell layout.

Accordingly, the optimal layout results in a reduction in the cycle time of 24.65 s compared to the T&N layout, which is an improvement of 10.2%. Nevertheless, it can be noted that the total area of the optimal layout is greater than the area of the T&N layout, as spaces appear between some of the machines.

The near-optimal layout, shown in Fig. 15, may be a very good solution in the event of additional constraints not considered in this study. Indeed, such constraints may show the inadequacy of the optimal layout, and the near-optimal layout might become, in such a case, a new optimal solution.

n°	1	2	3	4	5	6	7	8	9	10
1	49	3	12	12	12	11	12	11	12	11
2	3	2	0	0	0	0	0	1	0	0
3	12	0	6	0	0	0	0	0	0	0
4	12	0	0	6	0	0	0	0	0	0
5	12	0	0	0	6	0	0	0	0	0
6	11	0	0	0	0	6	0	0	0	0
7	12	0	0	0	0	0	6	0	0	0
8	11	1	0	0	0	0	0	6	0	0
9	12	0	0	0	0	0	0	0	6	0
10	11	0	0	0	0	0	0	0	0	6

Fig. 13. The rank table.

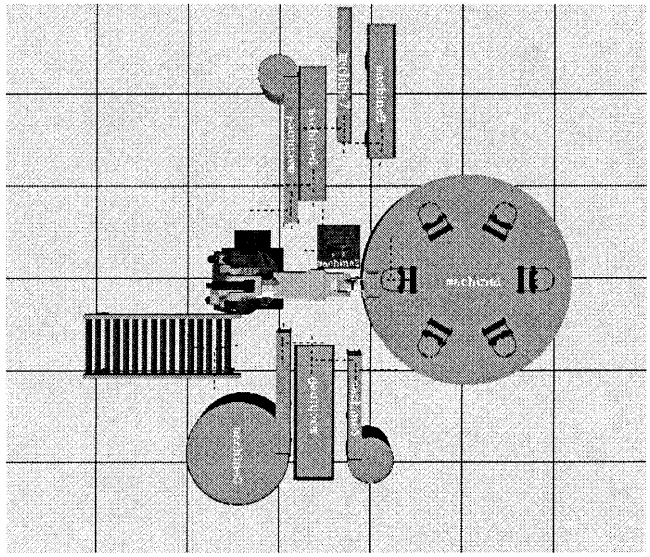

Fig. 14. The optimal layout generated by the program.

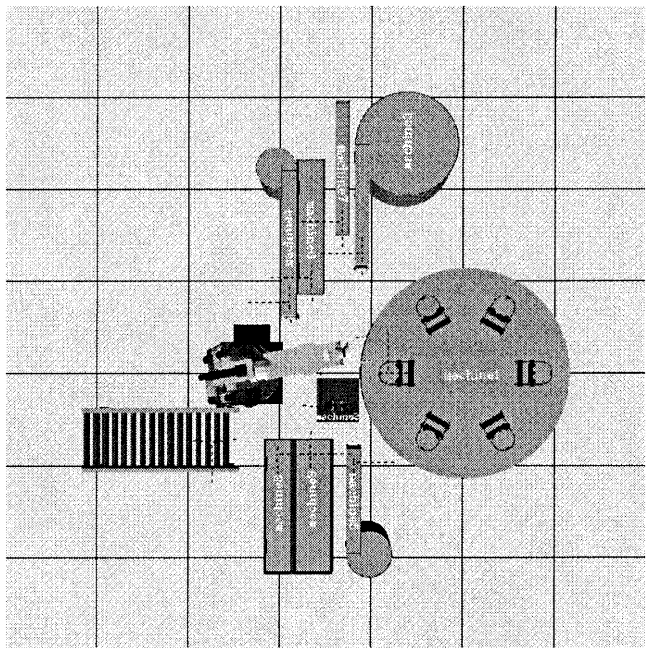

Fig. 15. Another layout generated by the program.

The computation time corresponding to the algorithm proposed is much greater than T&N. Nevertheless:

1. It is compatible with the use of the algorithm in the context of an industrial CAD/CAM/CAE system.
2. Several layouts are obtained.
3. The search space associated with the problem, i.e. the search space containing all the possible layouts, is explored widely, unlike the T&N method. Then, the computation time

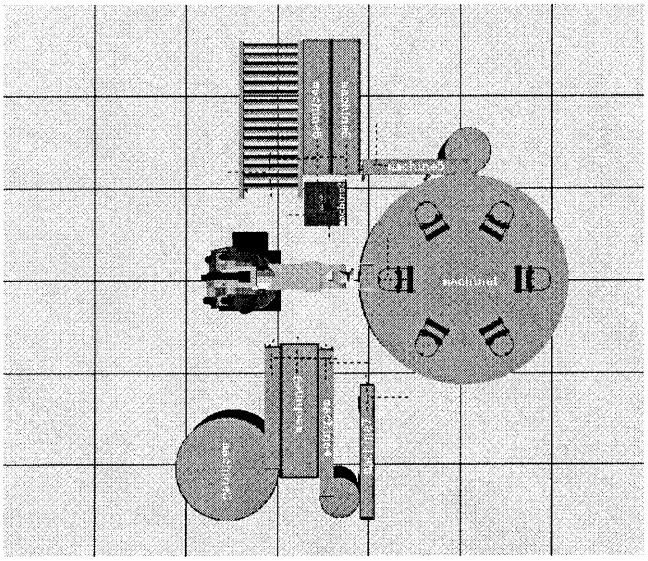

Fig. 16. The T&N layout.

Table 1. Cycle times.

	Optimal layout	Near-optimal layout	T&N layout
Cycle time (s)	216.7	219.25	241.35

Table 2. Computation times.

	Presented algorithm	T&N algorithm
CPU time (s)	282.6	55.8

Table 3. Cycle times.

	Mean (s)	Standard deviation (s)
Optimal layout	214.22	2.64
Worst near-optimal layout	216.31	1.89

obtained is thereby a larger proportion of the time necessary to reach an optimal solution of an assembly workcell layout problem.

6.3 Statistical Results

As mentioned previously, the method is stochastic. Hence, statistical results are essential to study its performance. The program was executed 50 times in succession. Table 3 gives means and standard deviations of the cycle times. "Optimal layout" means, in fact, "best layout of the five obtained", and "worst near-optimal layout" means, "worst layout of the five obtained". Table 4 corresponds to computation time.

Table 4. Computation time.

Mean (s)	Standard deviation (s)
276.23	8.72

The results obtained in the previous section are confirmed by these results. Note that the algorithm proposed yields on average an improvement of 11.2% compared to the T&N method, and that the best layout obtained resulted in a cycle time of 209.65 s.

7. Conclusion

In this paper, a method for optimising assembly workcell layout has been presented. Given a fixed sequence of operations, peripheral machines are positioned around the robot performing the task, with the aim of minimising the cycle time. The algorithm combines a constructive approach with a modified simulated annealing method. For the example used, this hybrid algorithm gave an important improvement in the cost function, together with other near-optimal layouts.

The major contribution of this work is that the problem is solved using an industrial CAD/CAM/CAE software product, IGRIP, with few restrictive working hypotheses. Applying the method yields several reliable optimal solutions to complex industrial cases.

References

1. P. Chedmail and P. Wenger, "Design and positioning of a robot in an environment with obstacles using optimal search", IEEE Transactions on Robotics and Automation, pp. 1069–1074, 1989.
2. P. Wenger, Design of Robotic Cells: Synthesis and Application, pp. 225–234, Kluwer, 1997.
3. S. K. Chittajallu and H. J. Sommer III, "Layout design for robotic assembly workcells", Technical Paper SME, AD86–409, pp. 7–59–7–69, 1986.
4. T. C. Lueth, "Automated planning of robot workcell layouts", Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1103–1108, 1992.
5. A. Kusiak and S. S. Heragu, "The facility layout problem", European Journal of Operational Research, 29, pp. 229–251, 1987.
6. M. L. Tay and B. K. A. Ngoi, "Optimising robot workcell layout", International Journal of Advanced Manufacturing Technology, 12(5), pp. 377–385, 1996.
7. Z. Drezner and S. Y. Nof, "On optimizing bin packing and insertion plans for assembly robots", IIE Transactions, 16(3), pp. 262–270, 1984.
8. S. Sahni and T. Gonzalez, "P-complete approximation problem", Journal of the ACM, 23, pp. 555–565, 1976.
9. R. E. Buckard and K. H. Stratmann, "Numerical investigations on quadratic assignment problems", Naval Research Logistics Quarterly, 25, pp. 129–148, 1978.
10. M. S. Bazaraa and O. Kirca, "A branch-and-bound based heuristic for solving quadratic assignment problem", Naval Research Logistics Quarterly, 30, pp. 287–304, 1983.
11. M. Scriabin and R. C. Vergin, "A cluster-analytic approach to facility layout", Management Science, 31(1), pp. 33–49, 1985.
12. K. Y. Tam, "Genetic algorithms, function optimization, and facility design", European Journal of Operational Research, 63(2), pp. 322–346, 1992.

13. A. Smith and D. Tate, “Genetic optimization using a penalty function”, Proceedings of the International Conference on Genetic Algorithms, pp. 499–505, 1993.
14. K. Kado, P. Ross and D. Corne, “A study of genetic algorithms hybrids for facility layout problems”, Proceedings of the International Conference on Genetic Algorithms, pp. 498–505, 1995.
15. Y. Kim, Y. Jang and M. Kim, “Stepwise-Overlapped Parallel Annealing and its Application to Floorplan Designs”, Computer-Aided Design, 23(2), pp. 133–144, 1991.
16. K. Y. Tam, “A simulated annealing algorithm for allocating space to manufacturing cells”, International Journal of Production Research, 30(1), pp. 63–87, 1992.
17. C. Sechen, “Chip-planning, placement, and global routing of macro/custom cell integrated circuits using simulated annealing”, Proceedings of the IEEE Design Automation Conference, pp. 73–80, 1988.
18. J. Cagan, D. Degenstehn and S. Yin, “A simulated annealing-based algorithm using hierarchical models for general three-dimensional component layout”, Computer-Aided Design, 30(10), pp. 781–790, 1998.
19. D. Barral, J.-P. Perrin, E. Dombre and A. Liégeois, “Development of optimisation tools in the context of an industrial robotic CAD software product”, *International Journal of Advanced Manufacturing Technology*, 15(11), pp. 822–831, 1999.
20. R. I. Chang and P. Y. Hsiao, “VLSI circuit placement with rectilinear modules using three-layer force-directed self-organizing maps”, IEEE Transactions on Neural Networks, 8(5), pp. 1049–1064, 1997.
21. B. K. A. Ngoi and K. Whybrew, “A fast spatial representation method”, International Journal of Advanced Manufacturing Technology, 8(2), pp. 71–77, 1993.
22. S. Kirkpatrick, C. D. Gelatt, Jr. and M. P. Vecchi, “Optimization by simulated annealing”, Science, 20(4598), pp. 671–680, 1983.
23. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, “Equations of state calculation by fast computing machine”, Journal of Chemical Physics, 21, pp. 1087–1091, 1953.
24. C. Sechen, VLSI Placement and Global Routing Using Simulated Annealing, Kluwer, Boston, 1988.
25. J. Hoshizaki, Robot Application Design Manual, pp. 284–287, Wiley, New York, 1990.