INT. J. COMPUTER INTEGRATED MANUFACTURING, 2000, VOL. 13, NO. 4, 301–310

Motion planning for multi-robot assembly systems

M. BONERT, L. H. SHU and B. BENHABIB

Abstract. The classical travelling salesperson problem (TSP) models the movements of a salesperson travelling through a number of cities. The optimization problem is to choose the sequence in which to visit the cities in order to minimize the total distance travelled. This paper presents a generalized point-to-point motion-planning technique for multi-robot assembly systems modelled as TSP-type optimization problems. However, in these augmented TSPs (TSP+), both the 'salesperson' (a robot with a tool) as well as the 'cities' (another robot with a workpiece) move. In addition to the sequencing of tasks, further planning is required to choose where the 'salesperson' (i.e., the tool) should rendezvous with each 'city' (i.e. the workpiece). The use of a genetic algorithm (GA) is chosen as the search engine for the solution of this TSP+ optimization problem. As an example area, the optimization of the electronic-component placement process is addressed. The simulation tools developed have been tested on five different component-placement system configurations. In the most generalized configuration, the placement robot meets the component delivery system at an optimal rendezvous location for the pick-up of the component and subsequently meets the printed-cirucit-board (on a mobile XY-table) at an optimal rendezvous location. In addition to the solution of the component-placement sequencing problem and the rendezvous-point planning problem, the collision-avoidance issue is addressed.

1. Introduction

Autonomous robotic systems are increasingly utilized in industrial environments, requiring the development of modelling methodologies and tools to optimize their operational efficiency. In this context, the classical assembly-planning optimization problem has been extensively studied, with numerous recent attempts at applying the earlier research results to robotic-based assembly (Drezner and Nof 1984). As expected, most proposed solution methods have their roots in the classical operations research (OR) field. Over the past several decades, OR research groups have devised many effective solution approaches to the combinatorial travelling salesperson problem (TSP), (Bozer *et al.* 1990), a two-dimensional point-to-point motion (PTP) optimization problem.

This paper presents a novel PTP motion-planning technique for multiple coordinated assembly robots, which can be modelled as a TSP, using genetic algorithms (GAs) (Goldberg 1953). As an example area, the optimization of the electronic component placement process is utilized (Chang *et al.*. 1987, Francis *et al.*. 1994).

1.1. *The travelling salesperson problem*

Many variations of this combinatorial problem have been addressed, including the asymmetric, symmetric, Euclidean, Chebyshev, prize collecting and time-dependent TSP variations (Dubowsky and Blubaugh 1989, Naft 1989, Bozer *et al.*. 1990, Ji *et al.*. 1992, Leu *et al.*. 1993, Huang *et al.*. 1994, Nelson and Wille 1995). For example, Leu *et al.*. (1993) solve the electronic-component placement optimization problem using genetic algorithms. The bin assignment problem is also addressed. Three types of assembly machines are modelled:

(*a*) a basic single robot pick-and-place problem with fixed feeders and a fixed printed circuit board (PCB)
(*b*) the above extended to have feeder component assignment optimization
(*c*) a multi-head fixed placement turret with a moving XY-table and feeder component optimization.

Authors: M. Bonert, L.H. Shu and B. Benhabib, Computer Integrated Manufacturing Laboratory, Department of Mechanical and Industrial Engineering, University of Toronto, 5 King's College Road, Toronto, ON, Canada, M5S 3G8; e-mail: beno@mie.utoronto.ca.

International Journal of Computer Integrated Manufacturing
ISSN 0951-192X print/ ISSN 1362-3052 online
http://www.tandf.co.uk/journals

1.2. *The augmented travelling salesperson problem (TSP+)*

In contrast to the single-robot TSPs, where the primary objective is to find the best sequence for N tasks, for multi-coordinated-robot problems one must also solve the rendezvous-point planning problem. In this augmented TSP (TSP+), the 'salesperson' as well as the 'cities' have motion capability. Namely, further planning is required to choose where the 'salesperson' should rendezvous with the 'city'.

There exist two variations to this rendezvous-planning problem: continuous-path planning where two robots move synchronously along a continuous curve (Suh and Shin 1987, Ahmad and Luo 1989, Tabarah *et al.*. 1994); and PTP path planning where the robots move independently and only rendezvous at discrete points. As one of the few research projects on PTP TSP+, Cao *et al.*. (1997, 1998) address the issue of inspection-task-sequence planning for two coordinated robots. Two SCARA robots are used to investigate the TSP+ problem, where one robot holds the inspection tool and the other holds the part to be inspected. A series of locations on a sphere are inspected by the robot pair, where both robots move together to bring the inspection tool and the part to the rendezvous location. Using the simulated-annealing technique, they were able to plan numerous point-to-point inspection routes. As expected, they showed that the cooperative robot configuration, where both robots moved, was faster than when only one robot moved while the other acted as a fixture.

1.3. *Augmented travelling salesperson problem with multiple robots*

If an additional (tool-carrying) robot is introduced into a TSP+ system, which continuously shares its workspace with the other (tool-carrying) robot, it would be necessary to directly address the collision avoidance problem. Such multiple-robot collision avoidance problems have been addressed in two primary ways in the literature: collision avoidance through path planning; and collision avoidance through scheduling (or time delays) (Baptiste *et al.*. 1992, Zurawski and Phang 1992, Baba and Kubota 1994, Chang *et al.*. 1994).

For example, Lee and Lee (1987) propose solving the collision-avoidance problem by speed changes, which introduce a time delay in one of the two robots. Each robot is represented by a single sphere at the wrist, and straight-line motion is assumed. For paths that are close together, collision maps of time versus distance travelled along the robot path are generated.

2. Problem definition and solution approach

In electronic assembly, components must be placed onto the PCB in a time-efficient manner. The first task is the configuration of the PCB, where component locations are determined subject to constraints and objectives. In this research it is assumed that this task has already been carried out. It is also assumed that the component-placement robot picks and places one component at a time. Although various other placement strategies exist, and are further detailed in the literature (Dubowsky and Blubaugh 1989), the objective of this paper is the investigation of the fundamental TSP+ problem as it applies to electronic component placement.

2.1. *Problem definition*

In this section, the problem will be defined for both the single- and two-robot cases.

2.1.1. *Set-up geometry.* Figure 1 shows the most generalized physical set up of the single-robot placement machine modelled. The system comprises four main sub-systems: an X-Y gantry robot for component pick-and-place operations; a numerically-controlled X-Y table, on which the PCB is located; and, two single-dof multiple-component delivery systems (CDSs), with controllable motions in the Y direction. For a two-robot placement machine, the system is identical to the single-robot system except for an additional X-Y gantry placement robot. The two gantry robots share a common workspace, which includes the workspace of their respective component-delivery systems and the workspace of the X-Y table. Each CDS is accessed by only the robot

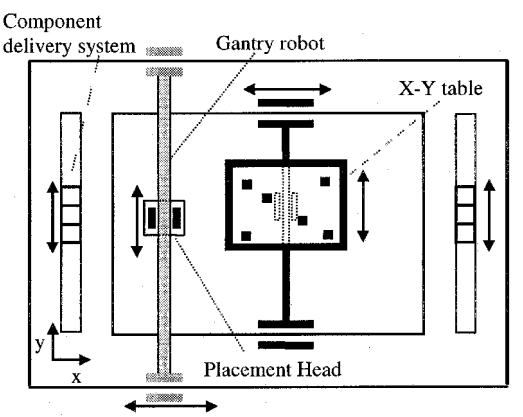

Figure 1. The single-robot electronic-component-placement machine configuration.

assigned to it. The two robots are not allowed to crossover each other.

2.1.2. *Problem structure.* For the single-robot configuration, there exist four sub-problems: assigning components to CDSs; determining the placement sequence; finding the rendezvous locations; and planning the device paths. Figure 2 shows the layers of the problem. The CDS assignment is the process of assigning the various component types to occupy specific bins in the two CDSs. The component placement sequence is a combinatorial optimization problem to minimize assembly time. The rendezvous-planning problem is the process of determining the meeting positions of the placement robot and the CDSs, and the meeting positions of the placement robot and the mobile X-Y table. When the combined degrees of freedom (dof) of the two coordinated subsystems is above the minimum needed, an infinite number of possible rendezvous-location solutions exist for every potential pick or placement exchange. Therefore, for a given sequence of picks and placements, a corresponding set of optimal rendezvous locations must be determined. In our case, for a given multi-component placement sequence, the optimality of a potential set of rendezvous locations can be determined by measuring the overall motion time. To achieve optimal results, individual robot paths between these rendezvous locations must also be optimized. This robot path sub-optimization problem is not addressed in this paper since it has been extensively addressed by the robotics research community (Cao and Dodds 1996). Herein, it is simply assumed that minimum robot-motion time can be achieved by minimizing the Cartesian distance travelled by the individual devices. For the two-placement-robot case, the problem has two additional layers of robot job assignment and collision avoidance. Although components are placed one at a time, according to an overall sequence, each component is placed by only one of the two robots. This requires each robot to be assigned the components it is responsible for placing. Since components are assumed to be placed sequentially, the placement sequence is identified as a variable. The third task is the solution of the rendezvous-location-planning problem for every pair of interacting mobile devices. Once the rendezvous positions have been determined, since there are two gantry robots sharing the workspace it is necessary to coordinate the actions of the two placement robots to prevent collisions. The final problem is the robot-path-planning problem. As per the single-robot problem, herein it is also assumed that minimum robot-motion time can be achieved by minimizing the distance travelled by the individual mobile devices.

2.1.3. *The assembly cycle.* In order to calculate placement cycle times, the overall board-population assembly problem is divided into individual cycles. Figure 3 shows the process of a generic pick-and-place cycle. The robot starts this cycle at its previous placement location and the CDS starts at its previous pick location. The CDS moves to the current pick location, Path #1, while the robot simultaneously moves there as well to rendezvous with the CDS, Path #2. The robot then picks the component from the CDS. While 1 and 2 are happening, the X-Y table moves to the current placement location, Path #3, where it meets with the robot arriving from the current pick location, Path #4. While 3 and 4 are happening, the CDS is allowed to move, without waiting, to the next pick location.. The robot then places the component and the cycle repeats.

2.2. *Proposed solution approach*

The goal of this paper is the development of a methodology to optimize the assembly process outlined above. Rather than optimizing one parameter at a time, a method that allows us to optimize all parameters simultaneously is chosen. This is advantageous, as many of the parameters are interdependent and the mod-

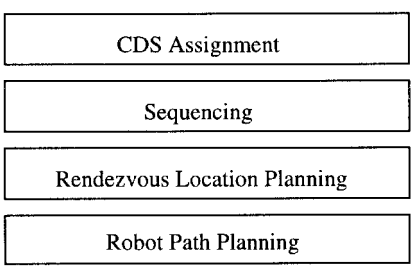

Figure 2. The various aspects of the single-robot problem.

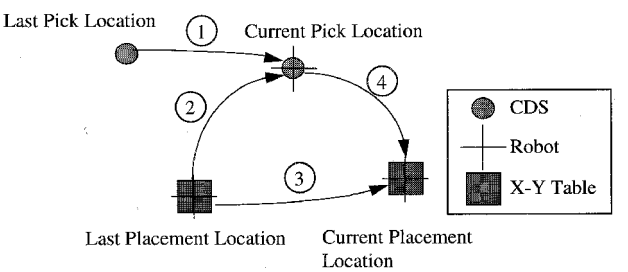

Figure 3. Illustration of a single pick-and-place cycle.

ification of one problem parameter affects another. The use of a GA was chosen as the solution approach for the problem at hand. The parameters for both the single- and two-robot placement problems are encoded into genomes (GA data strings). The fitness (objective function value) is used to determine which genomes are chosen for reproduction. The genetic operators generate new offspring, which are evaluated, and the entire process is repeated until the best genome is determined. To encode the single-robot parameters, a compound genome consisting of a series of sub-genomes is proposed: a sequencing sub-genome, two CDS allocation sub-genomes, and a rendezvous-point-planning sub-genome.

For the two-robot case, the solution strategy must also consider the collision avoidance between the two placement robots. As the parameters of sequencing and device positions during pick-and-place operations affect both the collision avoidance problem and the performance of a given placement strategy, it is advantageous to check both simultaneously when searching for an optimum solution. Therefore these parameters were integrated into a single objective function. To encode the two-robot parameters, a compound genome consisting of a series of sub-genomes is proposed: a sequencing sub-genome, two robot-assignment sub-genomes, and a rendezvous-point-planning sub-genome.

3. The single-robot problem (Bonert *et al.* 1998)

The goal of the single-placement-robot TSP+ problem is to minimize the total assembly time. The cycle time is used directly as the basis for the fitness score of a given GA genome. Although the solution of this problem was detailed by Bonert *et al.* (1998), it is briefly re-iterated here to clarify the rendezvous-point-planning problem for the two-robot case.

3.1. *Calculating cycle time*

As discussed in section 2, for a potential placement sequence the cycle time per component can be calculated based on the selected rendezvous locations between the devices and their initial locations. In order to convert this collection of positions into time, one needs to know the trajectory of each device from one point to the next. For simplicity, in this paper it is assumed that all devices move in straight lines between two points at devices' maximum allowable speeds. (Owing to the modularity of the proposed solution, one may utilize one of the many robot-trajectory optimization techniques proposed in the literature.)

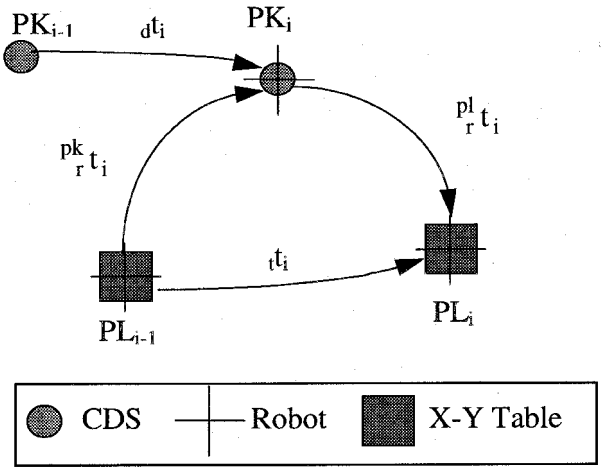

Figure 4. Illustration of cyclic device motion times.

From figure 4 one can see that there are four motion times that need to be calculated:

(*a*) the robot motion time-to-pick-location $^{pk}_r t_i$
(*b*) the CDS-motion-time to pick location $_d t_i$
(*c*) the robot motion time-to-placement-location $^{pl}_r t_i$
(*d*) the X-Ytable motion time-to-placement location $_t t_i$.

The overall assembly time t for a complete population of a PCB is calculated herein as follows:

$$t = \sum_{i=1}^{N} C_i \qquad (1)$$

where C_i is the time it takes to complete cycle i and N is the number of components on the PCB. One can note that the cycle C_i can be divided into two parts: the time before the end of the component pick operation, and the time after the component pick operation.

3.1.1. *The time before the end of the component-pick operation*. The limiting event is that both the robot and the corresponding CDS have to be at the pick location before the pick operation can proceed. Therefore, the maximum of the CDS and robot motion times to pick location must be considered as the overall pre-pick time. The robot motion time-to-pick-location with respect to the start of cycle C_i is simply $^{pk}_r t_i$. For the CDS, on the other hand, the portion of movement that occurs within cycle C_i is of interest. Thus, it is necessary to correct the CDS motion time to pick location $_d t_i$ by subtracting the

time the CDS has had off in the last cycle ${}^{off}_{d}t_i$. To complete the calculation of the first part of the cycle, the actual component-pick time ${}^{pk}C_i$ must be considered as well. Thus, the first part of the cycle time ${}^{pick}C_i$ is

$$ {}^{pick}C_i = max[{}^{pk}_{r}t_i, ({}_{d}t_i - {}^{off}_{d}t_i)] + {}^{pk}c_i \qquad (2) $$

In (2) the CDS off time (${}^{off}_{d}t_i$) is essentially the second part of the last cycle in which the CDS was utilized. Therefore, the second part of the cycle will be explained first to clarify the discussion of the CDS off time.

3.1.2. *The cycle after the component-pick-up operation.* This time is calculated with respect to the last known motion of the robot, which is the end of the ith component-pick operation. In order to determine the minimum time for the second part of the cycle, one has to determine the maximum of the time the robot takes to move to the placement location ${}^{pl}_{r}t_i$, and the time the X-Ytable takes to move to the same location ${}_{t}t_i$. The X-Ytable's motion-time-to-placement-location must consider the period that the table was previously off. In addition, the actual component-placement time ${}^{pl}c_i$ has to be added to the cycle time. Therefore, the second part of the cycle ${}^{place}C_i$ is

$$ {}^{place}C_i = max[{}^{pl}_{r}t_i, ({}_{t}t_i - {}^{off}_{t}t_t)] + {}^{pl}c_i \qquad (3) $$

In (3) the X-Y-table off time is equal to the entire first part of the cycle ${}^{pick}C_i$, namely

$$ {}^{off}_{t}t_i = max[{}^{pk}_{r}t_i, ({}_{d}t_i = {}^{off}_{d}t_i)] + {}^{pk}c_i \qquad (4) $$

Furthermore, in order to calculate the CDS off time in cycle C_i one has to determine the last time the CDS was picked from. If the CDS was used in the last cycle C_{i-1}, where the ith component is picked from the same CDS as the $(i-1)$th component, then the off time is equal to the second part of the previous cycle ${}^{place}C_{i-1}$:

$$ {}^{off}_{d}t_i = max[{}^{pl}_{r}t_{i-1}, ({}_{t}t_{i-1} - {}^{off}_{t}t_{i-1})] + {}^{pl}c_{i-1} \qquad (5) $$

Otherwise, if the component is picked from a different CDS, the off time must include the additional cycles the CDS had time off. Then, the overall off time is equal to the second part of the last cycle the CDS was picked from, plus all cycle times up to the current cycle that the CDS was off:

$$ {}^{off}_{d}t_i = \sum_{j=k+1}^{i-1} C_j + max[{}^{pl}_{r}t_k, ({}_{t}t_k - {}^{off}_{t}t_k)] + {}^{pl}c_k \qquad (6) $$

where the index k in the summation corresponds to the last component picked from the CDS under consideration.

The complete equation for a single cycle C_i is then the sum of the first part of the cycle ${}^{pick}C_i$ and the second part of the cycle ${}^{place}C_i$:

$$ C_i = max[{}^{pk}_{r}t_i, ({}_{d}t_i - {}^{off}_{d}t_i)] + max[{}^{pl}_{r}t_i, ({}_{t}t_i - {}^{off}_{t}t_i)] + {}^{pk}c_i + {}^{pl}c_i \qquad (7) $$

3.2. *Methodology*

With the above definition of the objective function and its various input parameters, it is possible to model any multi-robot coordinated system. The methodology adapted in this paper is the simultaneous solution of the multi-level optimization problem using a GA as the search engine. Namely, the optimal genome comprises the best placement sequence of the components as well as the best rendezvous locations between the robotic devices.

Since one of the goals of our work was to explore the effect of the introduction of more dof to an assembly system, our software is reconfigurable to run different set-ups: optimizing the positions of all-fixed devices; optimizing the fixed location of the X-Y table with a mobile CDS configuration; optimizing the fixed locations of the CDSs with a mobile X-Y table configuration; and, an all-devices-moving configuration. The software also allows users to either define the locations of all the components in the bins or have the GA optimize their locations.

3.3. *An example*

A simple PCB population sequence of six components is optimized. The general system set up described in section 2 is used. The system consists of one placement robot, two CDSs and an X-Y table. The two-dof gantry-type placement robot can move at a maximum speed of 2 m/s. The X-Y table can move at a maximum speed of 0.5 m/s. The CDSs move at a maximum speed of 1 m/s (Y axis).

For the system configuration where all devices can move and interact to yield the optimal assembly time, this simulation resulted in a total time of 1.525 s for a placement sequence of (0, 1, 2, 5, 4, 3). Figure 5 shows the robot pick-and-place path and the optimal rendezvous locations, for a user-defined CDS-component allocation case.

4. The two-robot problem

Section 3 presented the objective function for the single-robot TSP+ problem. This section will first define

the objective function for the two-robot case and then solve the TSP+ problem for the generic system configuration shown in figure 6. For clarity purposes, the collision avoidance issue will be addressed, in section 4.2, only after the sequencing and rendezvous-point-planning problems have been discussed in section 4.1.

4.1. *Problem solution without collision avoidance considerations*

4.1.1. *The objective function.* The objective function evaluates and assigns a fitness value to a genome. Assembly time is used directly as the basis for the fitness score. A genome for the two-robot case with no collision-avoidance consideration consists of four parts: the sequencing sub-genome, two robot-assignment sub-genomes, and the rendezvous-point sub-genome.

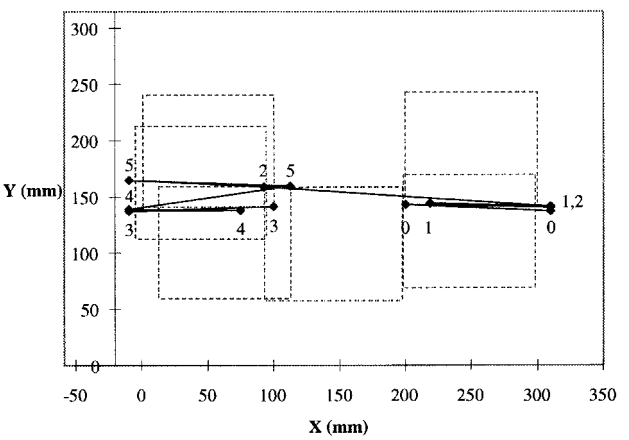

Figure 5. Robot pick-and-place path.

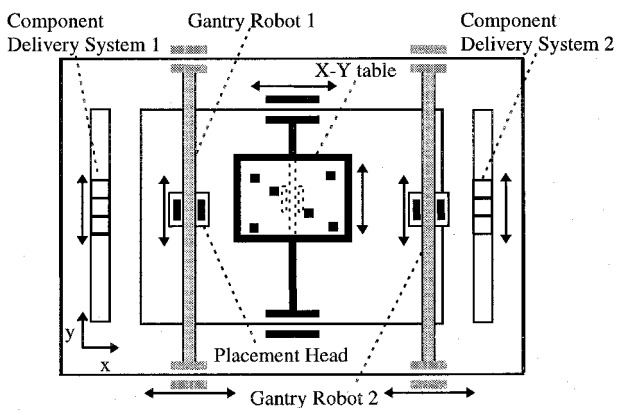

Figure 6. The two-robot electronic-component placement machine configuration.

For a given robot, the corresponding path determination procedure is similar to that of the single-robot problem. The robot path consists of motions between alternating pick-and-placement locations. These locations depend on the positions of the CDSs and the X-Y table. (i.e. rendezvous locations).

4.1.2. *Calculating motion times.* The cycle time C_i for two-robot system is defined by

$$C_i = max[{}_Rt_i, {}_tt_i] + {}^{pl}c_i \tag{8}$$

where the robot time ${}_Rt_i$ is the time in which the robot under consideration executes all tasks required and moves from the last placement location to be ready to place the next component at the current placement location. The X-Y table time ${}_tt_i$ is the time the table takes to move from the last placement location to the current placement location. The robot cycle time ${}_Rt_i$ can be divided into: the time before the pick operation, and the time after and including the pick operation.

The completion of the first part of the robot cycle time ${}_Rt_i^{1st}$ depends on both the CDS motion and the robot's motion to the pick location. Therefore, the first part of the robot cycle time is the maximum of the time the robot takes to reach the pick location and the time the CDS takes to reach it. The former is the robot motion time to pick location ${}_r^{pk}t_i$ minus the robot off time ${}_r^{off}t_i$. The latter is the CDS motion time to pick location ${}_dt_i$ minus the CDS off time ${}_d^{off}t_i$. Therefore

$${}_Rt_i^{1st} = max[({}_r^{pk}t_i - {}_r^{off}t_i), ({}_dt_i - {}_d^{off}t_i)] \tag{9}$$

The second term in (9) ${}_r^{off}t_i$ is the time the current robot has been off since its last placement operation. If the ith component is placed by the same robot as the (i-1)th component, then

$${}_r^{off}t_i = 0 \tag{10}$$

Otherwise, it is placed by the other robot and

$${}_r^{off}t_i = \sum_{j=k+1}^{i-1} C_j \tag{11}$$

where the index k represents the last cycle in which the current robot moved.

The fourth term in (9) ${}_d^{off}t_i$ is the time period that the current CDS has not been involved in a pick operation and has had time to move towards its next pick location. In order to calculate ${}_d^{off}t_i$, the total time of the last cycle in which the CDS was picked, Ck, is taken, and the time the CDS was busy is subtracted from it. The CDS is busy for the first part of the robot cycle time ${}_Rt_k^{1st}$, and the component pick time ${}^{pk}c_k$. If the ith component is picked from the same CDS as the (i-1)th component, then k=i– 1 and

$$ {}^{off}_{d}t_i = C_{i-1} - max[({}^{pk}_{r}t_{i-1} - {}^{off}_{r}t_{i-1}), ({}_{d}t_{i-1} - {}^{off}_{d}t_{i-1})] - {}^{pk}c_{i-1} \quad (12) $$

Otherwise, it is picked from the other CDS, where it is also necessary to add all the other cycles that the CDS has been off:

$$ {}^{off}_{d}t_i = \sum_{j=k}^{i-1} C_j - max[({}^{pk}_{r}t_k - {}^{off}_{r}t_k), ({}_{d}t_k - {}^{off}_{d}t_k)] - {}^{pk}c_k \quad (13) $$

where the index k is the last cycle in which a component was picked from the CDS under consideration.

The second part of the robot cycle time ${}_{R}t_i^{2nd}$ depends on the pick operation time ${}^{pk}C_i$ and the robot motion time to placement location ${}^{pl}_{r}t_i$ which occur sequentially

$$ {}_{R}t_i^{2nd} = {}^{pk}c_i + {}^{pl}_{r}t_i. \quad (14) $$

Adding the first and second parts of the robot cycle time, the overall robot motion time required in (8) is obtained as follows:

$$ {}_{R}t_i = {}_{R}t_i^{1st} + {}_{R}t_i^{2nd} = max[({}^{pk}_{r}t_i - {}^{off}_{r}t_i), ({}_{d}t_i - {}^{off}_{d}t_i)] + {}^{pk}c_i + {}^{pl}_{r}t_i. \quad (15) $$

4.2. *Two-robot problem with collision-avoidance considerations*

As discussed in section 1 the two basic collision-avoidance approaches noted in the literature are the path-planning-based approach and the time-delay-based approach. The collision-avoidance approach proposed here adds a safety delay to one of the two robots, delaying its motion until the other robot is no longer on its path. This method may appear to be an approach based on time delays; however, since collision avoidance is integrated into the GA (as described below) the end result is a combination of path changes and time delays.

For a given GA genome, potential collisions are averted by adding a safety delay to one of the two robots. Thus, the time it takes to complete the assembly for that particular configuration becomes longer (the fitness of the genome that is based on the assembly time becomes lower). This is where the correction for collision avoidance as part of the GA-optimization evaluation becomes valuable. Since the variables that affect the optimization of the system also determine whether collisions occur, a separate collision-avoidance correction carried out after the path-planning optimization could interfere with the result. Here, with the collision avoidance integrated into the optimization, as is possible with the GA, it is still possible to ensure that the solution is not only collision free but also optimized.

4.2.1. *The objective function.* The GA objective function evaluates a genome and assigns a fitness value to it. In this section the objective function is identical to the one presented in section 4.1.2 with an additional term related to collision avoidance.

4.2.2. *Calculating motion times.* The cycle time C_i is calculated using (8). However, the expression for the robot cycle time is modified by one additional safety-delay term ${}^{sd}_{r}t_i$:

$$ {}_{R}t_i = max[({}^{pk}_{r}t_i - {}^{off}_{r}t_i), ({}_{d}t_i - {}^{off}_{d}t_i)] + {}^{pk}c_i + {}^{pl}_{r}t_i + {}^{sd}_{r}t_i \quad (16) $$

Before the use of the safety-delay term in (16) one must determine if a collision is possible at all. It is assumed that the robots do not stop anywhere except at the pick and placement locations. Immediately after a placement operation, the current robot starts to move back to its pick location.

To carry out a collision check, the robot state must be modelled in space and time. In order to simplify the collision detection process, it was decided to model the robots as 'walls' moving across the workspace, as shown in figure 7. This modelling simplification reduces the state of the robot (for collision avoidance calculations) to two variables, the robot X-coordinate and a time at which the robot occupies the corresponding position.

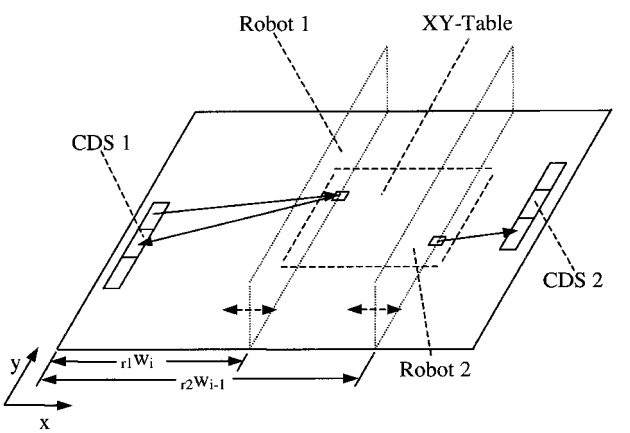

Figure 7. The robots modelled as walls for collision avoidance detection.

In figure 7 for the placement of a component Robot 1 moves from CDS 1 to the placement location. Its maximum travel toward the other robot is the X-coordinate of the placement location of component i, ${}_{r1}w_i$. The current robot (Robot 1 in the example) can collide with the other robot (Robot 2) that may still be in the common workspace. Therefore, in determining whether a safety delay is required, one must check all previous cycles, up to the last cycle, in which the current robot moved. Checking these previous cycles is necessary since the cycle sequence determines only when the components are placed, and not when the associated placement robot starts moving. With a large enough robot off-time, a robot in cycle i can start moving during cycle k+1, and reach the placement location (potentially in the other robot's path) to wait for the X-Y table.

For a collision to occur, the position and time for the two potentially colliding robots have to match. Therefore, the collision check for the current robot (in cycle i) checks all previous cycles until the current robot last move (cycle k). To check for an overlap, cycles (k+1) to (i– 1) are examined to ensure that the current robot placement location in the X axis (${}_{r1}w_i$) does not crossover with any of the other robot's previous placement locations (${}_{r2}w_{(k+1)}$ to ${}_{r2}w_{(i-1)}$). If an overlap occurs in one or more of the previous cycles, further testing to check the time variables is necessary to confirm or disprove a collision.

In this paper, collision is defined as a situation in which the current robot enters the overlap region before the other robot has left it. In the case in which both position and time variables indicate that the robots are in the overlap region at the same time, a preventive strategy must be developed. The approach here is to introduce a delay to the current (cycle's) robot's motion to allow the other robot to leave the overlap region before the current robot enters it.

After all the cycles j=(k+1) to (i– 1) have been tested for collision, it is possible to calculate the exact amount of time delay required to avoid a collision for a particular cycle j. The delay introduced is the time required by the obstructing robot to clear the overlap area before the current robot enters the overlap area (Bonert 1999).

4.3. *Simulation examples*

The following two configurations were considered for a six-component board:

(*a*) all devices moving with user-defined-CDS allocation; collision-avoidance routine is not employed

(*b*) all devices moving, with user-defined-CDS allocation; collision-avoidance routine is employed.

For case (*a*), the simulation yielded a minimum total time of 1.061 s with a placement sequence of (4, 2, 3, 0, 5, 1). Figure 8 shows the plot of the two robots' positions (X-coordinate) versus time, allowing for a quick visual collision identification. On such plots, any location where the line representing Robot 1's movement crosses Robot 2's line indicates a collision event. There are two collision events in the example considered. The first occurs when Robot 2 collides with Robot 1 while trying to place Component 2. The second collision occurs when Robot 2

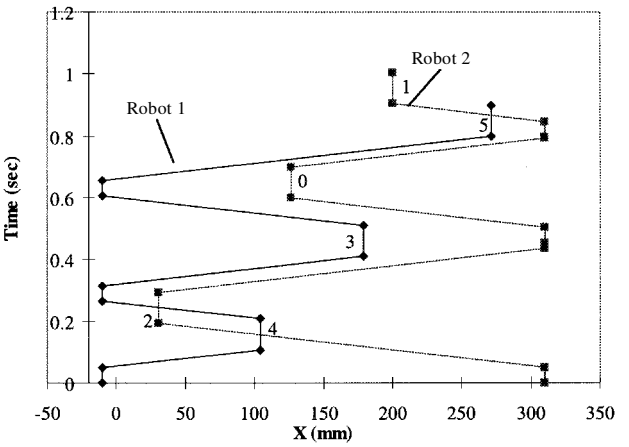

Figure 8. Robot X-coordinate versus time; no collision-avoidance routine.

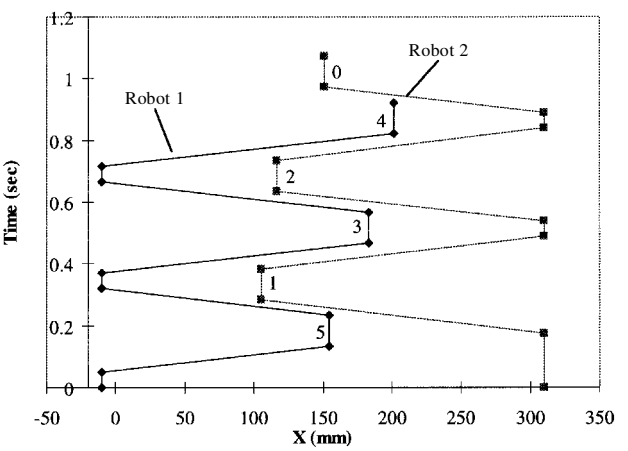

Figure 9. Robot X-coordinate versus time; with collision-avoidance routine.

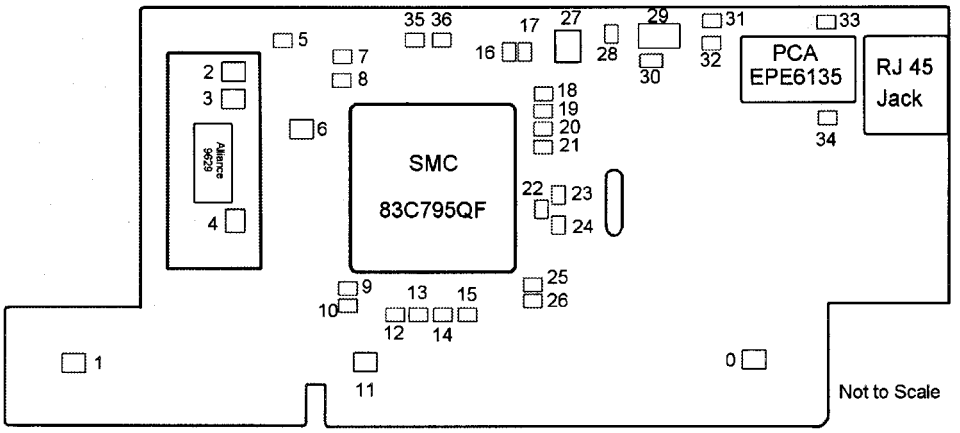

Figure 10. The SMC EtherEZ network card PCB.

tries to place Component 1 and collides with Robot 1 placing Component 5.

For case (*b*), where collision-avoidance is considered, the simulation yielded a minimum total time of 1.151 s with a different placement sequence of (5, 1, 3, 2, 4, 0). Figure 9 shows the robots' positions versus time. In this case, as expected, the robot time lines do not cross; no collisions occur.

The proposed methodology and the simulation tools were also used to plan an assembly process for a commercial PCB. The placement of all the surface mounted components on a network card shown in figure 10 was modelled.

The components were numbered from 0 to 36. The simulation for the single-robot case, (all devices moving with dynamic CDS allocation) resulted in an assembly time of 11.530 s with a placement sequence of (2, 35, 28, 14, 9, 12, 11, 21, 17, 32, 31, 10, 18, 36, 29, 1, 6, 8, 5, 3, 0, 25, 33, 34, 30, 7, 15, 24, 26, 22, 20, 4, 19, 27, 16, 13, 23). The simulation for the two-robot case resulted in a time of 6.123 s with a sequence of (33, 6, 32, 22, 29, 0, 19, 24, 3, 13, 20, 2, 11, 7, 15, 16, 9, 27, 14, 18, 21, 36, 10, 8, 4, 5, 35, 1, 17, 12, 31, 26, 28, 23, 30, 25, 34).

5. Conclusions

This paper has presented a generalized point-to-point (PTP) motion-planning technique for multiple coordinated assembly robots. The proposed approach minimizes assembly times using genetic algorithms (GAs). Specifically, as an example area the optimization of the electronic component placement process was addressed.

The collision avoidance strategy proposed is a rule-based approach and is directly incorporated into the objective function of the GA. Since the parameters of sequencing and device positions during pick-and-place operations affect both the collision avoidance problem and the performance of a given placement strategy, both are simultaneously optimized.

References

Ahmad, S. and Luo, S., 1989, Coordinated motion control of multiple robotic devices for welding and redundancy coordination through constrained optimization in Cartesian space. *IEEE Transactions on Robotics and Automation,* **5,** 409 – 417.

Baba, N. and Kubota, N., 1994, Collision Avoidance planning of a robot manipulator by using genetic algorithm: a consideration for the problem in which moving obstacles and/ or several robots are included in the workspace. *Proceedings of the First IEEE Conference on Evolutionary Computation*, New York, vol. 2, pp. 714 – 719.

Baptiste, P., Legeard, B. and Varnier, C., 1992, Hoist scheduling problem: an approach based on constraint logic programming, *Proceedings of the IEEE International Conference on Robotics And Automation,* Los Alamitos, vol. 2, pp. 1139 – 1144.

Bonert, M., Truillet, F., Shu, L.H, Sela, M.N and Benhabib, B., 1998, Optimal motion planning of robotic systems performing point-to-point electronic-assembly operations. *2nd World Automation Congress*, ISORA, Alaska, pp. 22.1 – 22.6.

Bonert, M., 1999, Motion planning for multi-robot assembly systems, MASc thesis, Department of Mechanical and Industrial Engineering, University of Toronto.

Bozer, Y. A., Schorn, E. C. and Sharp, G. P., 1990, Geometric approaches to solve the chebyshev traveling salesman problem. *IIE Transactions,* **22,** 238 – 254.

Cao, B. and Dodds, G. I., 1996, Implementation of near-time-optimal inspection task sequence planning for industrial robot arms. *Proceedings of the 4th International Workshop on Advanced Motion Control,* Mie, Japan, pp. 693 – 698.

Cao, B., Dodds, G. I. and Irwin, G. W., 1997, Time-suboptimal inspection task sequence planning for two cooperative robot arms using mixed optimization algorithms. *Proceedings of the IEEE International Conference on Robotics and Automation*, Albuquerque, pp. 2103 – 2108; 1998, A practical approach to near time-optimal inspection-task-sequence planning for two cooperative industrial robot arms. *International Journal of Robotics Research,* **17,** 858 – 867.

Chang, C., Chung, M. J. and Lee, B. H., 1994, Collision avoidance of two general robot manipulators by minimum delay time. *IEEE Transactions on Systems, Man and Cybernetics,* **24,** 517 – 522.

Chang, C., Smith, M. L. and Blair, E. L., 1987, Analysis of assembly planning problem for a robotized printed circuit board assembly center. *Proceedings of the 9th International Conference on Production Research, Cincinnati,* pp. 472 – 484.

Dubowsky, S. and Blubaugh, T. D., 1989, Planning time-optimal robotic manipulator motions and work places for point-to-point tasks. *IEEE Transactions on Robotics and Automation,* **5,** 377 – 381.

Drezner, Z. and Nof, S. Y., 1984, On optimizing bin picking and insertion plans for assembly robots, *IIE Transactions,* **16,** 262 – 270.

Francis, R. L., Hamacher, H. W. and Lee, C. Y., 1994, Finding placement sequences and bin locations for Cartesian robots. *IIE Transactions,* **26,** 47 – 59.

Goldberg, D. E., 1953, *Genetic Algorithms, in Search, Optimization and Machine Learning* (New York: Addison Wesley Publishing Company).

Huang, Y., Srihari, K., Adriance, J. and Westby, G., 1994, A solution methodology for the multiple batch surface mount pcb placement sequence problem. *Transactions of the ASME; Journal of Electronic Packaging,* **116,** 282 – 289.

Ji, Z., Leu, M. C. and Wong, H., 1992, Application of linear assignment model for planning of robotic printed circuit board assembly. *Transactions of the ASME; Journal of Electronic Packaging,* **114,** 455 – 460.

Lee, B. H. and Lee, C. S. G., 1987, Collision-free motion planning of two robots. *IEEE Transactions on Systems, Man and Cybernetics,* **17,** 21 – 32.

Leu, M. C., Wong, H. and Ji, Z., 1993, Planning of component placement/ insertion sequence and feeder setup in PCB assembly using genetic algorithm. *Transactions of the ASME; Journal of Electronic Packaging,* **115,** 424 – 432.

Naft, J., 1989, NEUROPT: Neurocomputing for multiobjective design optimization for printed circuit board component placement. *IJCNN: International Joint Conference on Neural Networks,* New York, NY, vol. 1, pp. 503 – 506.

Nelson, K. M. and Wille, L. T., 1995, Comparative study of heuristics for optimal printed circuit board assembly. *Southcon Conference Record,* pp. 322 – 327.

Suh, I. H. and Shin, K. G., 1987, Cooperative control of a robot and a positioning device. *IEEE Proceedings of the 1987 International Conference on Systems, Man, and Cybernetics*, New York, vol. 2, pp. 803 – 810.

Tabarah, E., Benhabib, B. and Fenton, R.G., 1994, Motion Planning for Cooperative Robotic Systems Performing Contact Operations Transactions of the ASME; *Journal of Mechanical Design,* **116,** 1177 – 1180.

Zurawski, R. and Phang, S., 1992, Path planning for robot arms operating in a common workspace. *Proceedings of the 1992 International Conference on Industrial Electronics, Control, Instrumentation, and Automation; Power Electronics and Motion Control,* New York, vol. 2, pp. 618 – 623.