OR Spectrum (2008) 30:535–549
DOI 10.1007/s00291-007-0088-1

REGULAR ARTICLE

A note on a motion control problem for a placement machine

Sofie Coene · Nguyen van Hop · Joris van de Klundert · Frits C.R. Spieksma

Published online: 7 July 2007
© Springer-Verlag 2007

Abstract Assembling printed circuit boards efficiently using automated placement machines is a challenging task. Here, we focus on a motion control problem for a specific type of placement machines. More specifically, the problem is to establish movement patterns for the robot arm, the feeder rack, and—when appropriate—the worktable, of a sequential pick-and-place machine. In this note we show that a (popular) greedy strategy may not always yield an optimum solution. However, under the relevant Tchebychev metric, we can model the problem as a linear program, thereby establishing the existence of a polynomial time algorithm for this motion control problem. Finally, we give experimental evidence that computing optimal solutions to this motion control problem can yield significantly better solutions than those found by a greedy method.

S. Coene (✉) · F. C. R. Spieksma
Operations Research Group, Katholieke Universiteit Leuven,
Naamsestraat 69, 3000 Leuven, Belgium
e-mail: sofie.coene@econ.kuleuven.be

F. C. R. Spieksma
e-mail: frits.spieksma@econ.kuleuven.be

N. van Hop
School of Advanced Technologies, Industrial Systems Engineering Program,
Asian Institute of Technology,
P.O. Box 4, Klong Kluang, Pathumthani 12120, Thailand

J. van de Klundert
Department of Mathematics, Maastricht University,
P.O. Box 616, NL-6200 MD Maastricht, The Netherlands
e-mail: j.vandeklundert@math.unimaas.nl

1 Introduction

Assembling printed circuit boards efficiently using automated placement machines is a challenging task. The ever increasing need for competitiveness means that improving the throughput of production lines is an important topic in this industry. It follows that investigating optimization problems for whole production lines as well as for individual machines remains a relevant task.

There are many types of different placement machines; for a more extensive discussion of different types of machines we refer to Grunow et al. (2004), and Egbelu et al. (1996), where a classification is proposed depending upon which parts of the machine can move. One possible categorization is to divide placement machines into two categories: sequential machines (machines in which each component is handled sequentially) and concurrent machines (machines in which components can be handled concurrently). For instance, machines featuring a rotating turret or carousel fall under the latter type. We restrict our attention here to sequential placement machines. This type of placement machines can be described as follows. It consists of three basic parts:

- a worktable. The worktable carries the printed circuit board and is able, in its most general form, to move in the x-direction and in the y-direction.
- a feeder rack. The feeder rack is a bar that contains feeders in which the components are stored. Notice that a feeder stores components of a single type. The feeder rack can move in the x-direction only.
- a robot arm. This is a device that transports the components from the feeder rack to the appropriate location above the board; it is able to move in the x-direction and in the y-direction.

Such a placement machine is described in, for instance, Ayob and Kendall (2005) (where the worktable can only move in the x-direction) and in Altinkemer et al. (2000) (where the worktable is stationary).

Now, in order to operate any placement machine, several decisions must be made. There are various hierarchies of decision making, see Crama et al. (2002) for a discussion of this subject. However, given a single machine and a single board, three basic problems need at least be addressed:

- the component sequencing problem: determine a sequence of the given locations on the board where the components will be placed,
- the feeder assignment problem: determine where the feeders are located in the feeder rack, and
- the component retrieval problem: determine for each component to be placed, from which feeder it will be retrieved.

Each of these problems has been studied extensively in the literature: early references to each of these three problems include Ball and Magazine (1988), Leipälä and Nevalainen (1989), Crama et al. (1996), we refer to Crama et al. (2002) and the references contained therein for more information concerning these problems.

In this note we focus on a motion control problem for a sequential placement machine as described earlier. Thus, we will assume that each of the problems mentioned

above has been solved. In addition, we assume (unless explicitly stated otherwise) that the robot arm can carry at most one component at any given moment in time. At first sight one may then wonder what is left to decide. However, before the machine starts actually inserting components, we need to establish the movement patterns of the three parts that are capable of moving: the robot arm, the feeder rack, and the worktable of the machine. This should be done in such a way that the machine finishes its last operation as soon as possible. We call this problem the motion control problem. Thus, the motion control problem for a single sequential pick-and-place machine that we address in this note can be described as follows. Given the locations on a board and a corresponding placement sequence, and given the location of each component in the feeder rack, the problem is to determine pick positions and place positions so that the last placement operation is executed as soon as possible. Our goal is to minimize the total assembly time for a single board.

Of course, for some sequential machines this problem is nonexistent. Indeed, if both the feeder rack and the worktable cannot move, movement of the robot arm is completely dictated by the solution to the three problems described above. Also, if the machine's technology is such that it features a fixed pick position and a fixed place position (i.e., each component is picked (placed) at the same prescribed position) the movement of the robot arm easily follows, as well as the movement of the worktable and the feeder rack. However, the motion control problem becomes interesting when there are no fixed pick and place positions, and at least two of the three parts are capable of moving. Indeed, in Su et al. (1995), a so-called *dynamic* pick and place model is introduced in which the possibility of dynamic pick and place positions is investigated.

Notice that we do not address a specific placement machine; to achieve competitive advantages, the technologies for pick-and-place operations are subject to constant change and refinement. Instead, our results apply to any (hypothetical) sequential placement machine featuring multiple moving parts. More generally, any situation where a transporting device needs to bring a set of items able to follow some movement pattern to a given set of locations also able to follow some movement pattern, falls under our scope.

1.1 Related literature

As mentioned, the problem of finding good operational solutions for a single placement machine has been actively investigated in literature. The motion control problem is first described in Su et al. (1995) who take into account the possibility of not restricting the pick positions and the place positions to given locations. They propose a greedy strategy to solve the resulting motion control problem and give computational evidence for the gain of this dynamic pick and place model compared to the setting with fixed pick and place positions. Further studies, that also involve the computation of a feeder assignment and a component placement sequence, are presented in Su and Fu (1998), Su et al. (1998), Wang et al. (1998), and Van Hop and Tabucanon (2001b). Van Hop and Tabucanon (2001a) and Ayob and Kendall (2005) each further develop a method for the motion control problem based on dynamic pick and place positions.

Motion planning has also received significant attention from the field of robotics (see e.g., Latombe (1991) for an overview). Here the emphasis is often on finding a motion plan (or a *path*) for a robot in some environment. Also, there is some literature that deals with classical routing problems such as the TSP, where the clients to be visited are known to move, and the salesman needs to take this into account, see for instance Helvig (2003), and the references contained therein. Asahiro et al. (2005), inspired by an application in robot navigation, deal with a similar problem, a variant of the *Vehicle Routing Problem* where moving elements need to be grasped one by one before they move out of the reachable region of the robot arm. The goal is to pick as many elements as possible. As far as we are aware however, the complexity of the specific motion control problem discussed here has not been answered before.

1.2 Our contribution

(i) We provide an example in which it is beneficial for a (moving) feeder rack to wait, thereby postponing the next picking moment. This example shows that GREEDY methods (see Sect. 2) do not always yield an optimal solution to the motion control problem (even in the case of a stationary worktable). The example is valid for each distance metric $(dx^p + dy^p)^{1/p}$ with $p > 0$ (where dx (dy) is the distance traveled in the x (y) direction), more specifically, the example is valid for $p = 1$ (the Manhattan norm), for $p = 2$ (the Euclidean norm), and for $p = \infty$ (the Tchebychev norm). Notice that the latter norm is quite common for placement machines.

(ii) We show that the motion control problem is solvable in polynomial time for the relevant Tchebychev norm by formulating the problem as a linear program.

(iii) We demonstrate that for randomly generated instances there is a significant difference between optimal solutions and solutions found by GREEDY methods. This difference partly depends on the ratio of the speed of the robot arm and the feeder rack. For some of the instances we considered the quality of a solution found by a GREEDY method may be significantly worse than the value of the optimum.

1.3 Remark

In our attempt to model the moving parts of a placement machine, we make assumptions that are not precisely fulfilled in practice. For instance, we assume a constant speed for each moving part; hence, we do not account for effects resulting from acceleration, and de-acceleration. For a description of the technical issues related to operating a placement machine we refer to Van Gastel et al. (2004).

2 A problem description, a method, and an instance

In this section we further describe the problem, and we sketch a class of solution methods for the motion control problem that we call GREEDY methods. Recall that

we assume that the component sequencing problem, the feeder assignment problem, and the component retrieval problem have been solved. In other words, the input to the motion control problem consists of (i) a sequence of n locations on the board, (ii) the position of the corresponding components in the feeder rack, and (iii) the starting configuration as well as the speeds of the robot arm, feeder rack, and worktable.

To facilitate the problem description we assume in this section that times needed for picking a component and times needed for placing a component can be ignored (notice that it is not difficult to include nonnegative picking and placing times in our methods and models, see Sects. 3 and 4). We assume that all movements occur in two-dimensional space, and hence, a position is completely specified by its x-coordinate and its y-coordinate. Also, we assume that the feeder rack coincides with the x-axis, i.e., all y-coordinates of picking positions equal zero. Finally, in order to facilitate the description of a GREEDY method, we first assume here that there are no physical obstructions for the movement patterns of robot arm, feeder rack, and worktable; we will come back to this issue later. We use the following notation:

for $i = 1, 2, \ldots, n$:

- (xp_i, yp_i): i-th placement position, i.e., the position where component i is placed by the robot arm onto the board,
- $(xs_i, 0)$: i-th pick position, i.e., the position where component i is picked by the robot arm from the feeder rack, and
- t_i^{place} (t_i^{pick}): moment in time when component i is placed (picked).

A feasible solution to the motion control problem amounts to finding values for these variables that correspond to achievable movement patterns. Further we use the following notation:

for $i = 1, 2, \ldots, n$:

- $(xb_i(t), yb_i(t))$: the position of the location on the board where component i needs to be placed at time t,
- $(xf_i(t), yf_i(t))$: the position of the location where component i is stored in the feeder rack at time t, and
- V_a (V_f, V_b): speed of the robot arm (feeder rack, board).

We call a method for the motion control problem a GREEDY method when, given the moments in time when the previous events occurred, the next event occurs as soon as possible. There are $2n + 1$ ordered events in the motion control problem: picking component i, placing component i ($i = 1, \ldots, n$), and returning to the starting position for the robot arm.

To describe a GREEDY method, let us for the moment assume that, for some i, $1 \leq i < n$, we know the i-th placement position (xp_i, yp_i), the corresponding time t_i^{place}, and that we also know the $(i + 1)$-st pick position $(xs_{i+1}, 0)$, and its corresponding time t_{i+1}^{pick}. Observe that we then also know $(xb_{i+1}(t_i^{\text{place}}), yb_{i+1}(t_i^{\text{place}}))$ and $(xf_{i+2}(t_{i+1}^{\text{pick}}), yf_{i+2}(t_{i+1}^{\text{pick}}))$, i.e., the position of the location where the $(i + 1)$-st component needs to be placed at time $t = t_i^{\text{place}}$, and the location in the feeder rack of component $(i + 2)$ at time $t = t_{i+1}^{\text{pick}}$.

Given the i-th placement position, and its corresponding time, and given the $(i+1)$-st picking position, and its corresponding time, we now show how a GREEDY method

computes the $(i+1)$-st placement position, the $(i+2)$-nd picking position, as well as their corresponding times. Applying this computation, starting with a given initial state, for $i = 1, 2, \ldots, n-1$ iteratively, gives us a solution to the motion control problem. This is done as follows. At $t = t_i^{\text{place}}$, we let the worktable move such that the location of the $(i+1)$-st placement location travels towards the $(i+1)$-st picking position. There are two possibilities. Either the board location arrives at the $(i+1)$-st picking position $(xs_{i+1}, 0)$ on or before $t = t_{i+1}^{\text{pick}}$, i.e., the board location arrives there before the robot arm (recall that we, for the moment, ignore potential physical obstructions). In that case, the board stops and waits for the robot arm to arrive. It follows then that $(t_{i+1}^{\text{place}}, (xp_{i+1}, yp_{i+1})) = (t_{i+1}^{\text{pick}}, (xs_{i+1}, 0))$. Or, the board and its $(i+1)$-st placement location is unable to reach the $(i+1)$-st picking position before $t = t_{i+1}^{\text{pick}}$, and given the pick occurring at $t = t_{i+1}^{\text{pick}}$, a placement position and time are computed by having the robot arm and board travel directly towards each other. This determines $(t_{i+1}^{\text{place}}, (xp_{i+1}, yp_{i+1}))$. To express this in mathematical terms, let f be a function which takes as input two states, each state corresponding to an object, where a state is specified by (time, location, speed). The function f then outputs the time and the location where the two objects meet, provided they travel directly towards each other. Thus:

$$\begin{aligned}&(t_{i+1}^{\text{place}}, (xp_{i+1}, yp_{i+1}))\\ &\quad = f((t_{i+1}^{\text{pick}}, (xs_{i+1}, 0), V_a), ((t_i^{\text{place}}, (xb_{i+1}(t_i^{\text{place}}), yb_{i+1}(t_i^{\text{place}})), V_b)).\end{aligned} \tag{1}$$

Next, given $(t_{i+1}^{\text{place}}, (xp_{i+1}, yp_{i+1}))$, we compute a minimal t_{i+2}^{pick} as follows. At $t = t_{i+1}^{\text{pick}}$, the feeder rack location of component $(i+2)$ starts to move towards the position on the x-axis where the robot arm can reach component $(i+2)$ as quickly as possible after having placed the $(i+1)$-st component. We express this using a function g that takes as input two states, each state corresponding to an object, where a state is again specified by (time, location, speed). The function g then outputs the minimal time and the corresponding location where the two objects meet, given that the second object moves only in the x-direction. Thus:

$$\begin{aligned}&(t_{i+2}^{\text{pick}}, (xs_{i+2}, 0))\\ &\quad = g((t_{i+1}^{\text{place}}, (xp_{i+1}, yp_{i+1}), V_a), ((t_{i+1}^{\text{pick}}, (xf_{i+2}(t_{i+1}^{\text{pick}}), yf_{i+2}(t_{i+1}^{\text{pick}})), V_f)).\end{aligned} \tag{2}$$

Equations (1) and (2) show how the $(i+1)$-st placement position, and the $(i+2)$-nd picking position, as well as their corresponding times can be computed when knowing the i-th placement position, the $(i+1)$-st picking position and the corresponding times. By viewing the starting configuration as the 0-th placement position, and by computing $(xs_1, 0)$ and a minimal t_1^{pick} given the starting configuration, we have specified a GREEDY method. We refer to the example below, and the corresponding Figure for an illustration of a GREEDY method.

Notice that we have not specified the precise form of the functions f and g; they depend on the particular distance metric used. We use f to find the meeting place, and

meeting time for two objects that each can move in both the x and y-direction, and we use g when one of the two objects can only move horizontally. Thus, in case the board is restricted to move only in the x-direction (see Ayob and Kendall (2005)), this is easily accommodated by replacing f by g in (1). Notice further that we assumed in the description above that there are no physical constraints for any of the moving parts. These constraints, however, are present in existing placement machines (although one could envision a technology where the feeder racks are above the board, and the board could travel beneath the rack without any physical constraints). Then, since the board should not collide with the rack, the board will not be able to reach a picking location. In the case these physical constraints play a role, we let a moving part (e.g., the work table) travel to the location that is *closest* to the location (under the appropriate norm) that was aimed for in case of the absence of these constraints. Observe also that, in case of the Tchebychev norm, there may be multiple locations each of which achieves a minimal time. We come back to this issue in Sect. 4. Finally, notice that GREEDY has the property that at any moment in time the robot arm moves (apart from the time spent in picking and placing the components).

Let us now proceed by sketching an example that shows that a GREEDY method may not always give an optimal solution.

Example Let us consider the following instance where we use the Tchebychev metric. The speed of the robot arm (denoted by V_a) equals 4 (measured in distance-units per time-unit), the speed of the feeder rack (denoted by V_f) equals 1, and the speed of the board (denoted by V_b) equals 0 (i.e., the board is stationary in this example). Let us assume that picking times and placing times can be ignored. Suppose further that at time $t = 0$ the robot arm is positioned at $(0, 0)$ and that it has to place two identical components that are stored in the feeder rack, currently positioned at $(20, 0)$. The first component has to be placed at $(20, 1)$ and the second at $(20, 2)$. The example instance is depicted in Fig. 1. The starting configuration of the instance is as follows: $t_0^{\text{place}} = t_0^{\text{pick}} = 0$, $(xs_0, ys_0) = (xp_0, yp_0) = (0, 0)$, $V_a = 4$, $V_f = 1$ and $V_b = 0$. Applying GREEDY to this instance yields the following:

$t = 4$: The robot arm meets the first component at $(16, 0)$, and picks it up. Using terminology introduced above, this is computed as follows:

$$(t_1^{\text{pick}}, (xs_1, 0)) = g[(t_0^{\text{place}}, (xp_0, yp_0), V_a), (t_0^{\text{pick}}, (xf_1(t_0^{\text{pick}}), yf_1(t_0^{\text{pick}})), V_f)].$$

Making g explicit we compute xs_1 and t_1^{pick} as follows:

since $yp_0 < |xp_0 - xf_1(t_0^{\text{place}})|$

$$\begin{aligned} xs_1 &= xf_1(t_0^{\text{place}}) - (|xp_0 - xf_1(t_0^{\text{place}})|/(V_a + V_f)) * V_f \\ &= 20 - (|0 - 20|/(1 + 4)) * 1 = 16 \end{aligned}$$

and

$$t_1^{\text{pick}} = t_0^{\text{pick}} + |xf_1(t_0^{\text{pick}}) - xs_1|/V_f = 0 + |20 - 16|/1 = 4.$$

Thus

$$g[(0, (0, 0), 4), (0, (20, 0), 1)] = (4, (16, 0)).$$

$t = 5$: The robot arm reaches the first placing location, and places the first component at (20, 1):

$$\begin{aligned}&(t_1^{\text{place}}, (xp_1, yp_1))\\ &\quad = f[(t_1^{\text{pick}}, (xs_1, 0), V_a), (t_0^{\text{place}}, (xb_1(t_0^{\text{place}}), yb_1(t_0^{\text{place}})), V_b)].\end{aligned}$$

Making f explicit we compute t_1^{place} and (xp_1, yp_1) as follows:

$$t_1^{\text{place}} = t_1^{\text{pick}} + \max(|xs_1 - xp_1|/V_a, yp_1/V_a) = 4 + (|16 - 20|/4, 1/4) = 5$$

and, as the board is stationary,

$$(xp_1, yp_1) = (xb_1(t_0^{\text{place}}), yb_1(t_0^{\text{place}})) = (20, 1).$$

Thus

$$f[(4, (16, 0), 4), (0, (20, 1), 0)] = (4 + (20 - 16)/4, (20, 1)) = (5, (20, 1)).$$

$t = 5.6$: The robot arm meets the second component at (17.6, 0), and picks it up:

$$\begin{aligned}(t_2^{\text{pick}}, (xs_2, 0)) &= g[(t_1^{\text{place}}, (xp_1, yp_1), V_a), (t_1^{\text{pick}}, (xf_2(t_1^{\text{pick}}), yf_2(t_1^{\text{pick}})), V_f)]\\ &= g[(5, (20, 1), 4), (4, (16, 0), 1)] = (5.6, (17.6, 0)).\end{aligned}$$

$t = 6.2$: The robot arm places the second component at (20, 2):

$$\begin{aligned}&(t_2^{\text{place}}, (xp_2, yp_2))\\ &\quad = f[(t_2^{\text{pick}}, (xs_2, 0), V_a), (t_1^{\text{place}}, (xb_2(t_1^{\text{place}}), yb_2(t_1^{\text{place}})), V_b)]\\ &\quad = f[(5.6, (17.6, 0), 4), (5, (20, 2), 0)]\\ &\quad = (5.6 + (20 - 17.6)/4, (20, 1)) = (6.2, (20, 2)).\end{aligned}$$

$t = 11.2$: The robot arm arrives at (0, 0).

Summarizing, robot arm and feeder meet for the first time at $t = 4$ at position (16, 0) where the picking of the first component occurs. The robot arm then travels to the first placing position (20, 1) and places the first component at $t = 5$. The picking of the second component takes place at time $t = 5.6$ at position (17.6, 0) and the placing at $t = 6.2$ at position (20, 2). Finally, the arm needs another 5 time units to return to (0, 0) such that total assembly is finished at $t = 11.2$.

Notice what would happen if we, starting with the initial configuration at $t = 0$, let the feeder rack move only 1 distance unit and wait with the feeder rack in (19, 0) for

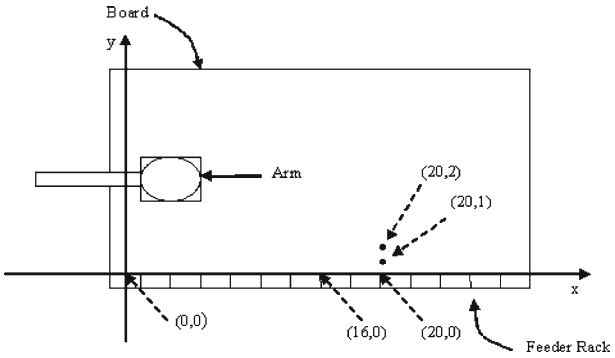

Fig. 1 Graphical representation of the example

the robot arm to arrive: then at $t = 4.75$, the robot arm would pick its first component at (19, 0), place this component at time $t = 5$, return to (19, 0) to pick the second component and place it at $t = 5.75$ and finally return to arrive at (0, 0) at time $t = 10.75$, which is faster than GREEDY's solution. Indeed, in this setting it is beneficial to wait with the feeder rack instead of moving it (one also can exhibit examples in which it is beneficial to wait with the robot arm instead of the feeder rack). The idea behind this example is that postponing the picking moment can actually decrease the time from place point to next place point. This can happen when the robot arm moves faster than the feeder rack. In this case it may be advantageous to travel with the robot arm only, instead of traveling with the both of them. More generally, when the speeds of two moving parts differ, GREEDY may not always find an optimal solution. Thus, intuitively, it can be better to use the "fast" moving piece and wait with the "slow" moving piece of equipment instead of moving them both.

Obviously, we do not claim that this is a realistic, or a worst-case example; the sole purpose of this example is to illustrate that GREEDY may not yield an optimum solution.

3 LP formulation

In this section we show that the motion control problem is solvable in polynomial time by formulating the problem as a linear program under the relevant Tchebychev metric. We assume (without loss of generality) that the rack has y-coordinate 0, and that all other y-coordinates are nonnegative; we also assume positive speeds for each of the moving elements. Further, we start from a situation (at $t = 0$) where the robot arm is located at (0, 0), and we impose that the robot arm has to return to (0, 0) after all components have been placed. We now state all variables and parameters we need to describe the model. We use the following variables, for $i = 1, \ldots, n$:

- xs_i: x-coordinate of the pick position of component i,
- xp_i: x-coordinate of the place position of component i,
- yp_i: y-coordinate of the place position of component i,

- T_i^1: time between picking component i and placing it,
- T_i^2: time between placing component i and picking component $i+1$.

(Notice that we let T_n^2 correspond to the time the robot arm needs between placing component n and returning to $(0, 0)$.) Finally, let

- T_0: time needed before picking component 1.

We use the following parameters:

- V_a: speed of the robot arm,
- V_b: speed of the worktable,
- V_f: speed of the feeder rack.

Further, for each component i to be placed ($i = 1, \ldots, n$) we have:

- pk_i: time needed to pick component i,
- pc_i: time needed to place component i.

Also, for any pair of locations i and $i+1$ to be visited consecutively ($i = 1, \ldots, n-1$), let

- dx_i: be the difference in x-coordinate,
- dy_i: be the difference in y-coordinate,
- d_i: be the difference (in x-coordinate) between the feeder from which component i is retrieved and the feeder from which component $i+1$ is retrieved.

Finally, let

- d_0: the distance (at $t = 0$) between $(0, 0)$ and the feeder holding the first component,
- (x_1, y_1): the x, y-coordinates of the location of the first component (at $t = 0$), and
- $xs_{n+1} = 0$.

$$\text{(MCP) Minimize} \quad T_0 + \sum_{i=1}^{n} (T_i^1 + T_i^2) \tag{3}$$

$$\text{subject to} \quad T_i^1 \geq \tfrac{yp_i}{V_a} \quad \text{for } i = 1, \ldots, n; \tag{4}$$

$$T_i^1 \geq \tfrac{|xs_i - xp_i|}{V_a} \quad \text{for } i = 1, \ldots, n; \tag{5}$$

$$T_i^2 \geq \tfrac{yp_i}{V_a} \quad \text{for } i = 1, \ldots, n; \tag{6}$$

$$T_i^2 \geq \tfrac{|xs_{i+1} - xp_i|}{V_a} \quad \text{for } i = 1, \ldots, n; \tag{7}$$

$$pc_i + T_i^1 + T_i^2 \geq \tfrac{|xs_i + d_i - xs_{i+1}|}{V_f} \quad \text{for } i = 1, \ldots, n-1; \tag{8}$$

$$pk_{i+1} + T_i^2 + T_{i+1}^1 \geq \tfrac{|xp_i + dx_i - xp_{i+1}|}{V_b} \quad \text{for } i = 1, \ldots, n-1; \tag{9}$$

$$pk_{i+1} + T_i^2 + T_{i+1}^1 \geq \tfrac{|yp_i + dy_i - yp_{i+1}|}{V_b} \quad \text{for } i = 1, \ldots, n-1; \tag{10}$$

$$T_0 \geq \frac{|d_0 - xs_1|}{V_f} \tag{11}$$

$$T_0 \geq \frac{|xs_1|}{V_a} \tag{12}$$

$$pk_1 + T_0 + T_1^1 \geq \frac{|x_1 - xp_1|}{V_b} \tag{13}$$

$$pk_1 + T_0 + T_1^1 \geq \frac{|y_1 - yp_1|}{V_b} \tag{14}$$

$$\text{all variables} \geq 0. \tag{15}$$

Notice that since the sum of all picking times and all placing times is a constant, the objective is formulated with (3). Constraints (4) imply that the time needed between picking component i and placing it (the left hand side) is at least equal to the time needed to travel with the robot arm in the y-direction to the y-coordinate of the next place position. In a similar fashion, constraints (5), (6) and (7) can be explained. Constraints (8) state that the amount of time needed between two consecutive picking operations (the left hand side) must be at least the time needed for the feeder rack to arrive at the position where the next component will be picked (notice that $xs_i + d_i$ reflects the position where component $i+1$ is at the time when component i is picked). Constraints (9) and (10) ensure that the board has enough time between two consecutive placement operations to arrive at the next placement operation. Finally, constraints (11)–(14) deal with the time needed for the first placement.

We make the following remarks:

- Strictly speaking, the model above is not a linear program due to the occurrence of absolute values. However, standard reformulation techniques can resolve this issue.
- This model can easily be modified for the case of a stationary worktable. Indeed, by dropping constraints (9) and constraints (10) and by turning the xp_i and yp_i from variables into parameters, we obtain a model for the case of a stationary worktable. Also, the model is easily adapted to deal with the case of a worktable being only able to move in the x-direction (see e.g., Ayob and Kendall 2005).
- Notice that in the description of this model we assume a single feeder rack. However, one easily generalizes this model to a setting where there are two feeder racks alongside the machine (or, even more general, when each component has its own specific travel characteristics, see Asahiro et al. 2005).
- In case there are limits for the robot arm, feeder rack, and board on the locations they can reach, one can add linear constraints ensuring these limits.

Finally, there are two important directions in which model (MCP) can be generalized. First, when a point is characterized by d coordinates (instead of two), the model can be easily adapted to deal with this situation. Second, in a setting where the robot arm has a capacity $c \geq 1$, the formulation remains valid. Indeed, it is not unnatural to assume that the robot arm can hold more than a single component (see e.g., Altinkemer et al. 2000), and as long as the sequence is specified with which these components need to be picked, and need to be placed, the model remains valid.

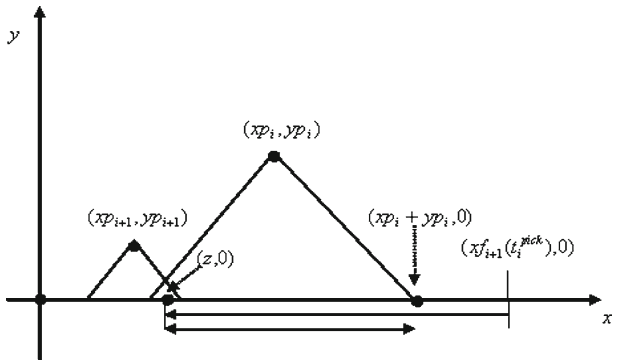

Fig. 2 GREEDY for the Tchebychev metric

4 Implementation, design, and computational results

4.1 Implementation and design

We first describe how we implemented a GREEDY method, and next, we discuss the design of the experiments.

As described in Sect. 2, a GREEDY method for the motion control problem consists in iteratively minimizing time between picking and placing a component and between placing a component and picking the next component. Indeed, suppose that the robot arm and the feeder rack meet each other somewhere on the x-axis to pick a component. From that point on the robot arm moves towards the placing position of that component (assuming a stationary worktable). The feeder starts moving at the same time with the next component to be picked in the direction of the next picking position. After placing, the robot arm returns to the x-axis and robot arm and feeder will meet as soon as possible. Because we are using the Tchebychev metric, this meeting point is not always uniquely determined, as is shown in Fig. 2. The robot arm can reach every point in the interval $[(xp_i - yp_i, 0), (xp_i + yp_i, 0)]$ in the same minimal timespan.

Suppose now that at $t = t_i^{\text{pick}}$, the feeder location of component $i + 1$ is to the right of $(xp_i + yp_i, 0)$, as indicated in Fig. 2. Suppose further that the feeder rack can reach up to $(z, 0)$ before the robot arm returns to the x-axis. It follows that every position in the interval $[(z, 0), (xp_i + yp_i, 0)]$ is a meeting point for robot arm and feeder rack achieving minimal time. In our implementation of a GREEDY method under a Tchebychev metric we choose as a meeting point the point which causes a minimal distance for the feeder rack to travel. Thus, we use as a secondary criterion the distance traveled by the feeder rack. In the example depicted in Fig. 2 this would amount to $(xp_i + yp_i, 0)$ as a meeting point.

Obviously, using knowledge of the next placement points may result in a better solution. Indeed, referring again to Fig. 2, given placement position (xp_{i+1}, yp_{i+1}), $(z, 0)$ may be a better meeting point than $(xp_i + yp_i, 0)$ when it comes to minimizing total time needed. However, we decided in our implementation of a GREEDY method

Table 1 Experimental design

Number of components	40/80/160
Number of component types	10/20
Length of the board	1,000 (in distance units)
Width of the board	500 (in distance units)
Length of the feeder rack	3,000 (in distance units)
Speed feeder rack/speed robot arm	0.25/0.5/1/2/4
Time needed to pick a component	0.8 (in time units)
Time needed to place a component	0.8 (in time units)
Capacity of head	1/4

not to use any information from upcoming placement positions, and instead use as a secondary criterion the feeder distance traveled.

The setting of our experiment is as follows: consider a board of length 1,000 and width 500 with n randomly generated placing positions on this board. We generated for m different component types a position on a feeder rack of length 3, 000. For each of the n components we uniformly selected a component type. The robot arm can move in the x- and y-direction, the feeder rack can move in the x-direction only, and the board is stationary. To completely specify an instance of the motion control problem, we took a random sequence of locations as the solution to the component sequencing problem, and we took a random assignment of feeders to positions in the rack as a feeder rack assignment. In addition, we assumed that there is precisely one feeder for each type of components, and hence, the component retrieval problem vanishes. As pointed out by a referee, the fact that these solutions are not found by some heuristic, may adversely affect the results of a GREEDY method, when compared to an optimum solution to the motion control problem.

Different experiments were executed by changing (i) the number of components to be placed ($n = 40, 80, 160$), (ii) the number of component types ($m = 10, 20$), (iii) the relative speeds of feeder and robot arm ($V_f / V_a = 0.25, 0.5, 1, 2, 4$), and (iv) the capacity c of the head of the robot arm ($c = 1, 4$); see Table 1 for an overview. In our choice for some of these parameter values, we used Van Gastel et al. (2004).

We implemented a GREEDY strategy in C^{++} language and we solved the LP's using ILOG CPLEX 8.1.0, OPL Studio 3.6.1. The tests were performed on a personal computer with a 2.8 GHz Intel(R)Pentium(R) IV with 504 MB of RAM. Since all computation times are within two seconds, we have not reported them.

4.2 Results

The results for the Tchebychev metric are summarized in Tables 2 and 3. Each number is the average over the results of ten different randomly generated instances.

Table 2 gives the percentage deviation $(= 100 \times \frac{\text{assembly time GREEDY} - \text{optimal assembly time}}{\text{optimal assembly time}})$ of the GREEDY heuristic from an optimal solution for a machine with a robot arm carrying at most one

Table 2 Percentage deviation of GREEDY from the optimum, head carries 1 component

n	m	V_f/V_a				
		0.25	0.5	1	2	4
40	10	5.599	5.843	7.971	16.218	20.398
80	10	6.512	6.675	8.108	14.674	19.965
160	10	6.769	6.735	8.194	16.895	21.690
40	20	7.436	7.114	8.544	16.941	20.552
80	20	6.984	6.825	8.149	16.602	20.457
160	20	7.046	7.240	8.544	16.018	19.636

Table 3 Percentage deviation of GREEDY from the optimum metric, head carries 4 components

n	m	V_f/V_a				
		0.25	0.5	1	2	4
40	10	1.470	1.791	2.469	5.624	7.668
80	10	1.844	2.069	2.600	5.862	7.566
160	10	1.861	1.990	2.616	6.352	7.988
40	20	1.476	1.754	2.100	5.621	7.501
80	20	2.021	2.281	2.563	6.549	8.182
160	20	1.773	1.983	2.375	6.132	7.470

component. It is clear that for a slow moving feeder rack (slow compared to the robot arm), the deviation of GREEDY's solutions from an optimal solution is relatively small. This is to be expected: in an extreme case of a stationary feeder rack, a solution found by a GREEDY method and an optimal solution coincide. However, GREEDY's performance deteriorates when the ratio V_f / V_a increases. Indeed, the slower the robot arm is (compared to the feeder rack) the larger the interval becomes where all meeting points have a minimal time between placing component i and picking component $i+1$. Also, the effect of the density (n/m) on GREEDY's performance seems relatively small.

In Table 3 the percentage deviation of GREEDY's solutions from optimal solutions is given when the head of the robot arm can carry at most four components, meaning that it can pick up four components before it travels to the board for placing. These results follow the same trend as the results in Table 2, namely a small deviation for a fast moving arm (compared to the feeder) which becomes larger as V_f / V_a increases. But, the percentage deviations in Table 3 are smaller than in the previous table. This can be explained by the fact that both a GREEDY solution and an optimal solution follow the same movement pattern during the time that the arm needs to pick up four components; only when the arm travels to the board to place the four components and then returns to the x-axis to pick the next component, differences may occur. And since the number of times the robot arm has to return to the feeder rack is now much smaller, the deviation of GREEDY's solutions compared to optimal solutions will be smaller.

5 Conclusion

We investigated the problem of how to determine movement patterns for the moving parts of an automated placement machine. We showed that a straightforward greedy strategy to establish these patterns may not give an optimal solution. However, under a realistic Tchebychev metric the problem is solvable in polynomial time by formulating it as a linear program. We showed that a reduction in assembly times is possible by using the LP-model.

Acknowledgments We thank a referee for the comments that led to an improved presentation of this note. This work was supported by the Research Foundation Flanders (FWO), grant G.0541.06.

References

Altinkemer K, Kazaz B, Köksalan M, Moskowitz H (2000) Optimization of printed circuit board manufacturing: integrated modeling and algorithms. Euro J Oper Res 124:409–421

Asahiro Y, Miyano E, Shimoirisa S (2005) Pickup and delivery for moving objects on broken lines. Lecture Notes in Computer Science 3701:36–50

Ayob M, Kendall G (2005) A triple objective function with a Chebychev dynamic pick-and-place point specification approach to optimise the surface mount placement machine. Euro J Oper Res 164: 609–626

Ball MO, Magazine MJ (1988) Sequencing of insertions in printed circuit board assembly. Oper Res 36:192–201

Crama Y, Flippo OE, van de Klundert JJ, Spieksma FCR (1996) The component retrieval problem in printed circuit board assembly. Int J Flexible Manufact Syst 8:287–312

Crama Y, van de Klundert JJ, Spieksma FCR (2002) Production planning problems in printed circuit board assembly. Discrete Appl Math 123:339–361

Egbelu PJ, Wu C, Pilgaonkar R (1996) Robotic assembly of printed circuit boards with component feeder location consideration. Prod Plann Control 7:162–175

Grunow M, Günther H-O, Schleusener M, Yilmaz IO (2004) Operations planning for collect-and-place machines in PCB assembly. Comput Ind Eng 47:409–429

Helvig CS, Robins G, Zelikovsky A (2003) The moving-target traveling salesman problem. J Algorithms 49:153–174

Latombe JC (1991) Robot motion planning. Kluwer, Boston

Leipälä T, Nevalainen O (1989) Optimization of the movements of a component placement machine. Eur J Oper Res 38:167–177

Su C, Fu H (1998) A simulated annealing heuristic for robotics assembly using the dynamic pick-and-place model. Prod Plann Control 9:795–802

Su C, Ho L, Fu H (1998) A novel tabu search approach to find the best placement sequence and magazine assignment in dynamic robotics assembly. Integ Manuf Syst 9:366–376

Su Y, Wang C, Egbelu P, Cannon DJ (1995) A dynamic point specification approach to sequencing robot moves for PCB assembly. Int J Comput Integ Manufact 8:448–456

Van Gastel S, Nikeschina M, Petit R (2004) Fundamentals of SMD assembly. Assembléon

Van Hop N, Tabucanon MT (2001a) Extended dynamic point specification approach to sequencing robot moves for PCB assembly. Int J Prod Res 39:1671–1687

Van Hop N, Tabucanon MT (2001b) Multiple criteria approach for solving feeder assignment and assembly sequence problem in PCB assembly. Prod Plann Control 12:735–744

Wang C, Ho L, Cannon DJ (1998) Heuristics for assembly sequencing and relatice magazine assignment for robotic assembly. Comput Indust Eng 34:422–431