Modelling the machine configuration and line-balancing problem of a PCB assembly line with modular placement machines

Journal:	*Engineering Optimization*
Manuscript ID:	GENO-2009-0162.R1
Manuscript Type:	Original Article
Date Submitted by the Author:	03-Nov-2009
Complete List of Authors:	Rong, Aiying; Technical University of Lisbon, Applied Mathematics and Economics Toth, Attila; University of Szeged, Juhasz Gyula Faculty of Education Nevalainen, Olli; University of Turku,, Information Technology Knuutila, Timo; University of Turku, Information Technology Lahdelma, Risto; Helsinki University of Technology, Energy Technology
Keywords:	Printed circuit board assembly, reconfigurable machine modules, line balancing

Modelling the machine configuration and line-balancing problem of a PCB assembly line with modular placement machines

Aiying Rong[1,4*], Attila Toth[2] , Olli S Nevalainen[1], Timo Knuutila[1], Risto Lahdelma[1,3]

[1]*Department of Information Technology and TUCS, University of Turku, 20014 Turku, Finland*
[2] *Juhasz Gyula Faculty of Education, University of Szeged, Szeged, Hungary*
[3] *Department of Energy Technology, Helsinki University of Technology, 02150 Espoo, Finland*
[4]*Cemapre (Center for Applied Mathematics and Economics), ISEG-Technical University of Lisbon, Rua do Quelhas 6, 1200-781 Lisboa, Portugal*

*Corresponding author, Tel: +351 213922747, Fax: +351 213922782
Email: aiying.rong@gmail.com

(Received 10 July 2009; final version received X X XXXX)

Abstract

Modular reconfigurable placement machines represent one of the most recent and most popular types of placement machines to respond to the needs for increased flexibility and productivity in automated printed circuit board (PCB) assembly. This paper studies the combined task of determining a favourable machine configuration and line balancing for an assembly line where a single type of PCB is assembled by a set of interconnected, reconfigurable machine modules. First, the problem is formulated as a non-linear integer programming model. Then it is transformed into a linear integer programming model that can be solved using a standard solver (e.g. Branch-and-Bound algorithm). The model determines the best machine configuration and allocation of components to the machine modules with the objective of minimizing the cycle time. Finally the effectiveness of the model is illustrated by numerical tests and the optimal solution from the model is compared with the result of a previous heuristic method.

Keywords: Printed circuit board assembly; reconfigurable machine modules; line balancing; integer programming; mixed integer linear programming.

1. Introduction

In modern electronics manufacturing, automated assembly systems are used to mount the electronic components at pre-specified locations onto printed circuit boards (PCBs). An automated PCB assembly line is typically laid out as a flow line of different types of machines. They are connected to each other by a conveyor system which transfers PCBs from one machine to another in the line. Each PCB has to visit all the machines before it exits the line (Ayob et al., 2008; Yilmaz et al., 2009).

The PCB assembly process consists of five major operations (Ho et al., 2008, Sze et al., 2001): application of adhesive or solder, component placement, reflow, cleaning, and testing & inspection. Among the assembly operations, the component placement is generally the most time-consuming (Ball and Magazine, 1988; Leipälä

and Nevalainen, 1989; Ashayeri and Selen, 2007). It is frequently a bottleneck of an assembly line and determines the line's cycle time (Ji et al., 2001).

There are three major tasks in planning and scheduling PCB assembly processes (Ammons et al., 1997): job grouping, line balancing (component allocation) and sequencing of placement operations. The first task is concerned with grouping various types of PCBs into families to reduce the setup time of the machines when moving from a product to another. The second task deals with the allocation of component types to machines and it defines the workload among the multiple machines of the assembly line when producing a single product. The third task is concerned with sequencing the placement operations within each machine to speed up the assembly. All these tasks are highly interrelated. However, each of them is in most cases very complex when formulated as a mathematical optimisation problem. It is therefore in practice impossible to solve these problems simultaneously.

This paper focuses on the joint problem of machine configuration and line balancing. The problem arises and has to be solved when an assembly line is configured and redesigned. Meanwhile, there is a great diversity in assembly machines employed in the industry. Ayob and Kendall (2008) classified the surface mount technology (SMT) placement machines into five categories based on machine types: dual-delivery, multi-station, turret-type, multi-head and sequential pick-and-place. *The reconfigurable modular machines* treated in this paper belong according to this classification to the type of multi-station machines and their modules are a kind of multi-head placement machines, also called collect-and-place machines (Grunow et al. 2004). Tirpak et al. (2002) provided a simulation toolkit for optimizing the PCB assembly lines based on a classification of SMT equipment.

The need for increased flexibility and productivity in PCB assembly has recently led to the development of modular reconfigurable placement machines, consisting of a number of small individual placement units. Besides offering a higher assembly speed and lower effort in material handling compared with conventional machine types, modular reconfigurable placement machines provide the possibility to change the setting of machine modules flexibly so that the machine configuration is suitable for the particular needs of the PCB-type to be processed.

Each reconfigurable module operates with the working principle of a collect-and-place placement machine, i.e. collective fetching and individual placements of the electronic components (Grunow et al., 2004, Ho et al., 2008). This paper focuses on the machine configuration and line balancing problem arising in such a setting. In this case, the line balancing problem comprises of the simultaneous machine configuration and component allocation among different machines to balance the workload. This problem is called the *machine configuration and line balancing* (MCLB) problem, see Toth et al. (2009) for heuristic solution of the problem.

This paper investigates a single PCB assembly line with a set of reconfigurable machine modules where a single type of PCB is assembled. Optimal configuration of a production line consisting of modular reconfigurable placement machine(s) is a difficult task. Each module is equipped with a stationary PCB holding table, stationary component feeder unit and a single moveable arm with a single changeable placement head. There are several different head types. Each of them may hold a certain set of component nozzles. As a PCB contains many components in different shapes, sizes and patterns, different component types require different nozzle types. The efficiency of the line depends on the combination of different modules equipped with different types of heads and nozzles. For each module, the compatibility of heads, nozzles and component types must then be considered. This

will give a number of constraints concerning the settings of a line of reconfigurable machine modules, making the MCLB problem much more complicated and harder to solve. When the components are mounted on the PCBs, all the modules are working concurrently. Therefore, the highest workload among the modules in the line determines the actual output rate, which is inversely related to the cycle time (i.e., the time for producing a single PCB). Thus, minimizing the cycle time can be taken as the objective of the problem. The cycle time is the maximum time that any of the machine modules needs to complete its assembly task on the PCB.

The main contribution of this paper is that the combined MCLB problem is formulated as a mathematical optimisation model that can in principle be used to determine the exact solution to the problem. The modular reconfigurable placement machines represent one of the most recent development and popular types of SMT assembly machines used in electronics industries (e.g. Fuji NXT or Siemens SIPLACE) to respond to the application's demanding high-speed chip placement, highly flexible end-of-line placement, or a combination of both. To our best knowledge, there is no such research work existing up to now because this machine type has been widely neglected in the academic literature. Due to high cost of placement machines, the optimisation of the assembly process can, no doubt, increase the manufacturing companies' competitiveness. Furthermore, under the mathematical framework, the problem can be described more rigorously and more accurately under certain mild assumptions. This facilitates understanding the problem better. Even though the line balancing problem in its simple form is already a NP-hard combinatorial optimisation problem (Gutjahr and Nemhauser, 1964), some exact solution approaches such as Branch-and-Bound (BB) algorithms have been used to solve certain variants of the line balancing problems (Baybars, 1986). In the present work small-size problem instances of the more complicated MCLB problem are managed to solve optimally by a standard software package. Using optimal solutions as references, the performance of heuristic algorithms can be evaluated. A less complex variant of the model can be used to find feasible (non-optimal) solutions. A known feasible solution can then be used to speed up both the BB search and various heuristic methods. In summary, the information provided in this paper is interesting and meaningful to both industrial practitioners and academic researchers.

The paper is organized as follows. Section 2 gives a brief overview of the relevant research on the topic. Section 3 presents a mixed integer linear programming (MILP) formulation for the MCLB problem. In addition to the actual MCLB optimization model, a simpler integer formulation for checking the feasibility of the machine configuration of PCB assembly line is also presented. The problem formulation of the MCLB problem is slow to solve by means of standard MILP optimizers. In order to speed up the solution process, Section 4 analyzes the problem structure and presents a heuristic approach based on the relaxation of optimization model presented in Section 3. Finally, Section 5 presents numerical results for a set of test problems.

2. Literature review

PCB assembly lines are flow oriented production systems, which have common characteristics of assembly lines in the industrial production of high quantity standardized commodities and low volume production of customized products. Among the optimisation problems which arise in managing such systems, assembly

line balancing is important for manufacturing companies to improve their productivity and minimize production costs (Becker and Scholl, 2006; Lapierre et al., 2006). The assembly line balancing problems are in most cases NP-hard (Gutjahr and Nemhauser, 1964). The line balancing problems in a general sense or in its basic form have attracted attention of academic researchers and industrial practitioners for many years (Amen 2006; Becker and Scholl, 2006; Scholl and Becker, 2006).

In PCB assembly, the line balancing problems appear when the component types are allocated to the machines in the line. One can classify the solution approaches for the line balancing problem into three categories: mathematical programming based approach (optimisation approach), tailored heuristics, and meta-heuristics.

Ammons et al. (1997) were among the first to investigate the problem of balancing the workload in PCB assembly systems by allocating the component types to the machines in the line. They formulated the problem as an integer programming (IP) model and applied both a heuristic approach and a standard research software package to solve the problem. Lin and Tardif (1999) investigated the PCB assembly line balancing considering the uncertainty in demand and capacity under the stochastic mixed integer programming framework. Then, they presented an approximate solution procedure based on solving the expected value model. The research direction of mathematical programming based approaches was further pursued by many other researchers (Lapierre et al., 2000; Hillier and Brandeau, 2001; Sze et al., 2001; Kodek and Krisper 2004). Sze et al. (2001) presented several mathematical models for the line balancing problem. Then the models were compared with some similar models to search an applicable algorithm. Lapierre et al. (2000) applied Lagrangian relaxation techniques to solve the problem. Kodek and Krisper (2004) developed an efficient Branch-and-Bound (BB) algorithm to handle the problem. Hillier and Brandeau (2001) used both the BB algorithm and a specialized heuristic algorithm to solve the problem.

Tailored heuristic approaches have often been used to solve the line balancing problem in PCB assembly. For example, Tazari et al. (2006) solved the problem by combing the network flow approach and shortest-path based multi-exchange local search. Häyrinen et al. (2000) and Yildirim et al. (2006) developed dispatching rules to allocate the component types to the machines in the assembly line. Choudhury et al. (2007) decomposed the process planning decisions into four related problems and proposed efficient heuristics to solve the problem. Yilmaz et al. (2009) applied some specialized heuristics under the simulation framework.

Application of meta-heuristics is also common. Ji et al. (2001) and Kulak et al. (2007) presented a genetic algorithm (GA) to allocate the placement operations to the machines in the line. Wan and Ji (2001) gave a tabu search heuristic approach. Khoo and Alisantoso (2003) proposed an immune algorithm. For reconfigurable modular machines, Toth et al. (2009, submitted) used an evolutionary algorithm (EA) to solve the problem.

3. Problem description and formulation

3.1 Machine configuration and line balancing problem

The combined machine configuration and line balancing (MCLB) problem consists of two interrelated decisions: determining the machine configuration and allocation of components to be placed by different machine modules.

<Figure 1 is around here>

Figure 1 illustrates the machine modules and their arrangement along the production line. A conveyor carries the PCBs from one module to the next. The different modules operate concurrently, but their working areas are fairly narrow. If the PCB is wide, this means that each module can only access a section of the PCB at a time, and it may be necessary to advance the conveyor during the assembly process.

Each module is equipped with a stationary PCB holding table, a stationary component feeder unit and a moveable arm with a multi-nozzle placement head. The component feeder unit holds a limited number of positions (slots) for component feeders. The component feeders may be of several different types, like tapes, sticks, etc. Each of the feeders holds components of a single type. Each component feeder occupies a certain number of slots on the feeder unit depending on the width of the individual components.

The reconfigurable placement machine module works similar to a collect-and-place machine (Grunow et al., 2004; Ho et al. 2008). The machine module operates in cycles of picking up and placing components on the PCB. During each cycle, the placement head travels first to the feeder unit and picks up a number of components from the component feeders one by one using nozzles attached to the head. Each nozzle can grasp one component on the same cycle. Then the placement head traverses to the PCB and places the components one by one at their predefined locations. The total time for these cycles depends on several factors including assignment of nozzles to head, placement ordering and feeder assignment etc. The cycle time is the sum of picking up and placing time plus the movement time of the placement head between the feeder unit and the PCB.

Reconfigurable machine design means that each machine module can be equipped with different head types, nozzle types and feeder units to match the particular PCB assembly task. Only one head is assigned to each module, but the head can carry multiple nozzles based on the compatibility between the head and nozzle types. The head can be equipped with different nozzle types or multiple copies of the same nozzle type. The present study omits the effect of the topology of the PCB and the ordering of the arrangement of the component feeders within the feeder unit. The study thus lets the processing time of a machine module be linearly dependent on the number of component placements and on the number of pick-and-place cycles performed by the module. This simplification is partially justified by the small physical dimensions of the modules and thus stresses the importance of the partitioning of the assembly tasks to different modules.

The compatibility between head types and nozzle types (meaning that a certain head type can handle a certain set of nozzle types), the component types and nozzle types (meaning a certain nozzle type can handle a certain set of component types), and the component types and head types (which transitively follows from previous two relations) are given by binary matrices. A value of 1 in the compatibility matrix means that the type represented by the row is compatible to the type represented by the column.

To facilitate modelling, the present study makes the following assumptions, similar to Toth et al. (2009, submitted).

- A component type can be handled by only one nozzle type.
- A nozzle type can be handled by only one head type.
- Enough copies of each head type and nozzle type are available.

The first assumption simplifies computation of the number of pick-and-place cycles and reduces the number of constraints in the original formulation. This assumption is not very restricting, because in reality, there is typically for each component type a primary nozzle type that is ideal for placing it. The primary nozzle type is applied whenever possible. Using a secondary nozzle type may impose quality problems in form of higher failure rates and lower quality. A secondary nozzle type is therefore used only if no feasible configuration using the primary nozzle can be found. The first and second assumptions together impose restriction on the module-head assignment because it follows transitively that a component type can be handled by only one head type. The corresponding head type that can handle the component type must be installed in the module once a component type is assigned to a module, The third assumption makes it possible to choose any head type for a module and any nozzle type for a head purely based on the compatibility of the matrix between head and nozzle types. However, it would be easy to extend our model with constraints on the number of copies of head and nozzle types.

In determining the component allocation, the important considerations include whether a particular component can be placed by a particular machine module based on its configuration, the width of the component, the capacity of the feeder unit attached to the machine module, the amount of time required by the machine module to pick and place the components, and the amount of time required by the machine module to perform the necessary moves between the feeder unit area and the PCB area. The last two time factors can only be estimated when the component allocation is solved independently of the other tasks as described in the introduction. Here these two time factors are assumed to be given.

3.2 Problem formulation

The following notations are given to formulate the MCLB problem.

Indices:

i head type,
j nozzle type,
l machine module, and
k component type.

Sets:

A component types,
H head types,
L machine modules, and
N nozzle types.

Parameters:

$B^{hn}_{i,j}$ 0-1 matrix stating whether head type i is compatible with nozzle type j,

$B^{an}_{k,j}$ 0-1 matrix stating whether component type k is compatible with nozzle type j,

D_k number of components of type k that need to be placed on the PCB,
E_k feeder width of component type k,
F_l width (capacity) of the feeder unit attached to module l,
G_i maximum number of nozzles that can be attached to head type i,
M a big positive number,
T_l^{pp} average pick-and-place time of module l for each component, and
T_l^{tr} average travelling time factor per pick-and-place cycle for module l performing the movements between the feeder unit and PCB.

Decision variables:

$x_{l,k}$ number of components of type k that are placed by module l,
$z_{l,j}$ number of nozzles of type j that are attached to module l ,
w_l number of pick-and-place cycles performed by module l ,
$y_{l,i}^h$ binary variable indicating whether head type i is attached to module l, and
$y_{l,k}^a$ binary variable indicating whether component type k is allocated to module l.

The MCLB problem can be formulated as follows.

$$\text{Minimize } \max_{l\in\mathbf{L}}\left(\sum_{k\in\mathbf{A}} T_l^{pp} x_{l,k} + T_l^{tr} w_l\right) \quad (1)$$

subject to

$$\sum_{i\in\mathbf{H}} y_{l,i}^h = 1,\ l\in\mathbf{L}, \quad (2)$$

$$z_{l,j} \le M\sum_{i\in\mathbf{H}} B_{i,j}^{hn} y_{l,i}^h,\ l\in\mathbf{L},\ j\in\mathbf{N}, \quad (3)$$

$$\sum_{j\in\mathbf{N}} z_{l,j} \le \sum_{i\in\mathbf{H}} G_i y_{l,i}^h,\ l\in\mathbf{L}, \quad (4)$$

$$y_{l,k}^a \le \sum_{j\in\mathbf{N}} B_{k,j}^{an} z_{l,j},\ l\in\mathbf{L}, k\in\mathbf{A}, \quad (5)$$

$$x_{l,k} \le M y_{l,k}^a,\ l\in\mathbf{L}, k\in\mathbf{A}, \quad (6)$$

$$\sum_{k\in\mathbf{A}} E_k y_{l,k}^a \le F_l,\ l\in\mathbf{L}, \quad (7)$$

$$\sum_{l\in\mathbf{L}} x_{l,k} = D_k,\ k\in\mathbf{A}, \quad (8)$$

$$\sum_{k\in\mathbf{A}} x_{l,k} B_{k,j}^{an} \le w_l z_{l,j},\ l\in\mathbf{L}, j\in\mathbf{N}, \quad (9)$$

$$y_{l,i}^h \in \{0,1\},\ l\in\mathbf{L}, i\in\mathbf{H}, \quad (10)$$

$$y_{l,k}^a \in \{0,1\},\ l\in\mathbf{L}, k\in\mathbf{A}, \quad (11)$$

$$x_{l,k} \in \{0,\ldots,D_k\},\ l\in\mathbf{L},\ k\in\mathbf{A}, \quad (12)$$

$$w_l \in \{0,\ldots,\max_k\{D_k\}\},\ l\in\mathbf{L}, \quad (13)$$

$$z_{l,j} \in \{0,\ldots,\max_i\{G_i | B_{i,j}^{hn}=1\}\},\ l\in\mathbf{L}, j\in\mathbf{N}. \quad (14)$$

The objective function (1) is to minimize the maximum processing time of the machine modules. The machine module with the maximum processing time is the *bottleneck module* that determines the output of the production system. The processing time of the machine module consists of the picking up and placing time of the components allocated to the machine module (the first term) and the travelling

time between the feeder unit and PCB (the second term). Constraints (2) mean that exactly one head is attached to each module. Constraints (3) state that one or more copies of a nozzle type can be assigned to a module only if it is compatible with the head assigned to the module. Constraints (4) mean that the total number of nozzles attached to a head cannot exceed the capacity of the head. Constraints (5) mean that a component can be allocated to a module only if it is compatible with at least one nozzle that has been attached to the module. Constraints (6) allow one or more copies of a component type to be placed by a module only if the component type has been allocated to the module. Constraints (7) state that the total width of the component types allocated to a module can not exceed the capacity of the feeder unit. Constraints (8) mean that all components of each type must be allocated to modules. Constraints (9) determine the number of pick-and-place cycles w_l for each module l. These inequality constraints state that for each nozzle type j the number of cycles must be large enough to allow the $z_{l,j}$ copies of the nozzle type to place out those allocated components that are handled by that nozzle type. (Recall the assumption that each component type is handled by a single nozzle type). Constraints (10)-(14) define the domains of the decision variables.

The objective function (1) and the constraints (9) introduce non-linearity in the model. Therefore, the above model formulation should be transformed into an equivalent linear model so that a standard software package can be used to solve it.

3.3 Linearization of the model

The objective function (1) can be linearized by standard techniques. By writing $\tau = \max_{l \in \mathbf{L}} \left(\sum_{k \in \mathbf{A}} T_l^{pp} x_{l,k} + T_l^{tr} w_l \right)$, objective (1) can be reduced to

$$\text{Minimize } \tau \tag{15}$$

$$\sum_{k \in \mathbf{A}} T_l^{pp} x_{l,k} + T_l^{tr} w_l \le \tau, \; l \in \mathbf{L}. \tag{16}$$

Linearization of the product of two variables in constraints (9) is more complicated. To do that, constraints (9) should be rewritten by counting values of $z_{l,j}$ sequentially, i.e. 0, 1, …. This produces sets of linear constraints. However, only one of these constraint sets should be active. To encode the activation and deactivation of the linear constraints, new binary variables $y^n_{l,j,c}$ need introducing. Variables $y^n_{l,j,c}$ indicate the c^{th} copy of nozzle type j is installed on module l for each $c \in \mathbf{C}_j$. Here, $\mathbf{C}_j = \{1, \ldots, \max_i \{G_i | B^{hn}_{i,j} = 1\}\}$ is the set of copies of nozzle type j. Constraints (9) can now be replaced by the following two sets of linear constraints:

$$\sum_{k \in \mathbf{A}} x_{l,k} B^{an}_{k,j} \le c w_l + M y^h_{l,j,c+1}, \; l \in \mathbf{L}, \; j \in \mathbf{N}, c = 1, \ldots, |\mathbf{C}_j| - 1, \tag{17}$$

$$\sum_{k \in \mathbf{A}} x_{l,k} B^{an}_{k,j} \le |\mathbf{C}_j| \, w_l, \; l \in \mathbf{L}, \; j \in \mathbf{N}. \tag{18}$$

Constraint (17) restricts the value of w_l only when exactly c copies of the nozzle are installed. If the next $(c+1)^{\text{st}}$ copy of the nozzle is present, then the M-term deactivates the constraint. If the $(c+1)^{\text{st}}$ copy of the nozzle is missing, then the constraint is dominated by the previous instance of the constraint (corresponding to c

installed nozzles). Constraint (18) treats the case corresponding to $c=|\mathbf{C}_j|$ copies of nozzle type j.

The new binary variables replace $z_{l,j}$ in the model. Constraints (3) are replaced by the following two sets of constraints.

$$y^n_{l,j,1} \le \sum_{i\in\mathbf{H}} B^{hn}_{i,j} y^h_{l,i},\ l \in \mathbf{L},\ j \in \mathbf{N}, \tag{19}$$

$$y^n_{l,j,c} \le y^n_{l,j,c-1},\ l \in \mathbf{L},\ j \in \mathbf{N},\ c = 2,...,|\mathbf{C}_j|. \tag{20}$$

Constraints (19) state that the first copy of the nozzle can be assigned to a module only if it is compatible with the head assigned to the module. Constraints (20) state that the c^{th} copy of a nozzle can be added only if the (c-1)st copy exists.

Constraints (4) and (5) are replaced by

$$\sum_{j\in\mathbf{N}} \sum_{c\in\mathbf{C}_j} y^n_{l,j,c} \le \sum_{i\in H} G_i y^h_{l,i},\ l \in \mathbf{L}, \tag{21}$$

$$y^a_{l,k} \le \sum_{j\in\mathbf{N}} B^{an}_{k,j} y^n_{l,j,1},\ l \in \mathbf{L}, k \in \mathbf{A}. \tag{22}$$

Finally, the domain of new variables are defined.

$$y^n_{l,j,c} = \{0,1\},\ l \in \mathbf{L},\ j \in \mathbf{N},\ c \in \mathbf{C}_j, \tag{23}$$

$$\tau \ge 0. \tag{24}$$

As a result of the above transformation, objective function (15) with constraints (2), (6)-(8), (10)-(13), and (16)-(24) form a standard mixed integer linear programming (MILP) model for the MCLB problem. The problem can be solved by a standard MILP solver. However, both the number of constraints and number of variables increase significantly due to this transformation.

3.4 Finding feasible solutions

Sometimes it is useful to find out if the MCLB problem is feasible or not, prior to starting the search for the optimal or even suboptimal solution. If it can be proven that no feasible solution exists, a more time-consuming optimisation process can be avoided. A known feasible solution provides an upper bound for the optimal solution, and during e.g. Branch and Bound search, this upper bound can be used to prune unpromising branches from the search tree. This can greatly speed up the search, in particular if the upper bound is close to the true optimum. Also several heuristic methods for solving the problem can benefit from a known feasible solution, either as a starting solution or as providing an upper bound for the optimum.

A simpler version of the MCLB problem can be formulated in order to find out if the original problem has a feasible solution. Infeasibility can be caused only by not having enough space in the feeder units to accommodate all component types or by not having enough space in the heads to allocate all required types of nozzles. The possibility of having multiple copies of a single nozzle in one head does not affect the feasibility of the problem. It may speed up the operation of a single machine by reducing the number of cycles. Also, the allocation of a certain component type on

multiple machines does not affect feasibility; however, it may speed up operation by balancing the workload.

The feasibility of the problem can therefore be detected by a relaxed model obtained by removing the objective function, and the equations related to the computation of the number of placements and cycles. In particular, this means

- omitting the w_l variables,
- replacing the variables $x_{l,k}$ by $y^a_{l,k}$
- replacing the variables $z_{l,j}$ by $y^n_{l,j}$ (binary variable indicating whether nozzle type j is attached to module l),
- omitting constraints (6) and (9), and
- setting $D_k = 1$.

As a result, the feasibility can be checked by solving the following set of constraints as an integer linear programming problem with an arbitrary objective function:

$$\sum_{i\in\mathbf{H}} y^h_{l,i} = 1,\ l \in \mathbf{L}, \tag{25}$$

$$y^n_{l,j} \le \sum_{i\in\mathbf{H}} B^{hn}_{i,j} y^h_{l,i},\ l \in \mathbf{L},\ j \in \mathbf{N}, \tag{26}$$

$$\sum_{j\in\mathbf{N}} y^n_{l,j} \le,\ l \in \mathbf{L}, G_i y^h_{l,i} \tag{27}$$

$$y^a_{l,k} \le \sum_{j\in\mathbf{N}} B^{an}_{k,j} y^n_{l,j},\ l \in \mathbf{L}, k \in \mathbf{A}, \tag{28}$$

$$\sum_{k\in\mathbf{A}} E_k y^a_{l,k} \le F_l,\ l \in \mathbf{L}, \tag{29}$$

$$\sum_{l\in\mathbf{L}} y^a_{l,k} = 1,\ k \in \mathbf{A}, \tag{30}$$

$$y^h_{l,i}, y^n_{l,j}, y^a_{l,k} \in \{0,1\},\ l\in\mathbf{L},\ i\in\mathbf{H},\ j\in\mathbf{N},\ k\in\mathbf{A}. \tag{31}$$

3.5 A sample MCLB problem

To facilitate understanding the structure of the MCLB problem, a simple numerical example is presented in Table 1 (Toth et al., 2009, submitted).

<Table 1 is around here>

4. Analyzing problem structure

Generally speaking, a standard MILP solver cannot handle combinatorial optimisation problems of Section 3 efficiently. Here the structure of the MCLB problem is briefly analyzed to find some ways to speed up the solution process. To facilitate this analysis, it is assumed that the travelling time factors T^{tr}_l are same for all of the machine modules, i.e., $\alpha = T^{tr}_l\ (l \in \mathbf{L})$. Then, parameter value $\alpha = 0$ means that the travelling time is totally ignored. When $\alpha \neq 0$ (>0), the value of "α" reflects the relative value of travelling time as compared with the pick-and-place-time.

4.1 Ignoring travelling time

When $\alpha = 0$, it is equivalent to ignoring the second term in objective function (1). The problem becomes easier and the size of the problem decreases significantly. Variable w_l and constraints (9) of the original formulation then become unnecessary. In the meanwhile, a general integer variable $z_{l,}$ reduces to a binary variable and M equals 1 in constraints (3) (see constraints (26)). This means that a single copy of each nozzle type is sufficient and all constraints in the model are linear. In fact, the feasibility test model presented in Section 3.4 is related to the case $\alpha = 0$. That is, the linearity transformation of the non-linear constraints is needed only when $\alpha \neq 0$. The objective function only introduces trivial non-linearity, which can be handled by the standard techniques as shown in Section 3. Consequently, the number of constraints and the number of decision variables decreases significantly as compared with the case for $\alpha \neq 0$.

Under the assumptions made in Section 3.1, since a component type can be handled by only one nozzle type and a nozzle type can be handled by one head type, it follows transitively that a component type can be handled by only one head type. Consequently, the solution structure of the case with $\alpha = 0$ bears some similarity to the case for $\alpha \neq 0$. Both the cases with $\alpha = 0$ and $\alpha \neq 0$ consider the balance of component allocation among the machine modules. This means that module-head assignment should be the same for $\alpha = 0$ and for $\alpha \neq 0$ because the corresponding only one head type must be installed in the module once a component type is allocated to a module. The difference for $\alpha \neq 0$ lies in the fact that more copies of some nozzle types are needed to handle some frequently used component types to reduce the number of pick-and-place cycles for a module.

Therefore, when the problem instance with $\alpha \neq 0$ is solved, one can fix the module-head assignment beforehand based on the solution for $\alpha = 0$. This can speed up the solution process significantly. In addition, when $\alpha = 0$, the solution can identify the potential bottleneck modules and give the rough number of components that the bottleneck modules should handle. It is possible that more than one module can become a potential bottleneck module. However, in a general situation, if a nozzle type can be handled by more than one head type, and a component type can be handled more than one nozzle type, then this may cause that a component type to be handled by more than one head type. Consequently, the head-module assignment for $\alpha = 0$ may not be same as that for $\alpha \neq 0$.

4.2 Partial model relaxation

When problem instances with $\alpha \neq 0$ are solved, one can relax the integer variables $x_{l,k}$ (the number of components of type k that are placed by module l) and w_l (the maximum number of pick-and-place cycles of module l) into continuous (real) variables. In most cases, such a relaxed problem is much easier to solve than the original integer programming problem. The solution of the relaxed problem serves as a lower bound for the original problem. This solution gives a rough number of components that the potential bottleneck nozzle handles and a rough number of components that the potential bottleneck module handles. This relaxed solution

maintains the feasibility of the module-head and nozzle-head assignment but the number of components assigned to each module may be a fractional number.

However, a straightforward round-up (down) process may not give a good feasible solution, especially when α is relatively large. When α is a small positive number, the pick-and-place time of the components accounts for a large portion of cycle time, and the amounts of the components allocated to the potential bottleneck modules are roughly same. However, for a large α, the travelling time between feeder unit and PCB would account for a large portion of the total processing time. Component allocation among the potential bottleneck modules tends then not to be as even as for smaller α. There is a trade-off between the travelling time and pick-and-place time. It is all possible that the bottleneck module would change if both $x_{l,k}$ and w_l are restricted to be integers. The component allocation among the potential bottleneck modules needs adjusting to get a good trade-off. For example, when the number of components is rounded to an integer, the changes of w_l for a module should be checked. Several trial-and-error adjustments are needed to get a suitable integer for both $x_{l,k}$ and w_l.

4.3 Heuristic solution approach

The above analysis gives us the following heuristic solution procedures for solving the original MCLB problem by solving the relaxed problem first.

Step 1. Solve the problem (15) with constraints (2), (6)-(8), (10)-(13) and (16)-(24) for $\alpha = 0$.

Step 2. Solve the relaxed problem of Section 4.2 with the fixed module-head assignment from Step 1.

Step 3. Restore the feasibility (i.e. integer requirements for the number of components and pick-and-place cycles) of the relaxed solution of Step 2.

5. Computational results

To evaluate the effectiveness of the model presented in Section 3 and the heuristic solution procedure of Section 4.3, a standard MILP software package is used to solve the small MCLB instances and one bigger instance reported in Toth et al. (2009, submitted). The test instances reflect the real life PCB instances in the proportional sense. Though the number of head types, number of nozzle types, and number of component types are smaller than those in reality, the proportion of them follows the real life PCB instances. The bigger instance approximates real life instances. Table 2 shows these problem instances and their sizes as MILP models (number of constraints ×number of decision variables). It is assumed that the travelling time factors $\alpha = T_l^{tr}$ ($l \in \mathbf{L}$) are the same for all of the machine modules. The MCLB problems are solved using five travelling time factors: α = 0, 1, 2, 5 and 10. α = 0 means that the arm travelling time is totally ignored. Parameter value α = 1 and 2 mean moderate relative travelling time. α = 5 and 10 mean large relative travelling time. The values of "α" affect the problem size as shown in Table 2. When α = 0, the problem size is much smaller than that for $\alpha \neq 0$ as discussed in Section 4. When $\alpha \neq 0$, the relative value of "α" can affect how the components are allocated among the machines. This also has an impact on the solution time of the solver (see Table 3).

For all these cases, the optimisation model was run in two modes. The first is the relaxation mode (R) as discussed in Section 4.2. The second is the true

optimisation mode (O) of the model presented in Section 3.3. For $\alpha = 0$, the results of the two modes are same. All the test runs were performed in a 2.79 GHz (1 GB RAM) Pentium 4 PC under the Windows XP operating systems.

<Table 2 is around here>

For both optimisation and relaxation modes, the solution process can be speeded up by first solving the problem instance with $\alpha = 0$ as discussed in Section 4.3. Table 3 gives the solution times of the optimisation modes O and R with $\alpha = 0$, 1, 2, 5 and 10. Table 4 gives the solutions (i.e. values of formula (15)) generated by optimisation mode (O) and relaxation mode (R) with $\alpha = 0$, 1, 2, 5 and 10 as well as the gap of the relaxed solution against optimal solution.

$$\text{GAP } (\%) = 100*(z_o - z_R)/z_o, \quad (32)$$

where z_R is the solution generated by the relaxation mode and z_o is the optimal solution. The solution time for the optimisation model with $\alpha \neq 0$ is the running time of the optimizer when module-head assignment is fixed based on the solution with $\alpha = 0$. The solution time can be improved from 2 to 5 times as compared with the situation when the solution process starts from the scratch.

<Table 3 is around here>
<Table 4 is around here>

Based on Table 3, the solution time for the optimisation model is sensitive to the problem size and travelling time factor α. The computational time is related to the size of the mathematical model (see Table 2) but not purely determined by the size. Other parameters such as α can affect solution time. The solution efficiency of the optimisation model with the two modes differs significantly. The solution time for the relaxation mode is much shorter than that for the optimisation mode if the problem is solvable. Generally speaking, for the same problem instance, the solution times for both modes O and R have tendency to increase as α increases (though not strictly monotonically). For $\alpha = 0$, the solution time is short. This means that the trade-off between pick-and-place time and travelling time becomes more difficult to manage as the travelling time factor α becomes large for both modes. Therefore, it is not surprising to see that it is even difficult to solve the relaxed version of the problem for $\alpha > 1$ of the problem instance 4. In addition, the solution time increases (though not strictly monotonically) as the problem size increases (shown in Table 2). Furthermore, it is easy to check whether the configuration of the modular system is feasible based on the feasibility test model in Section 3.4 because the structure of the feasibility test model is similar to the case $\alpha = 0$ but a little simpler.

Based on Table 4, GAP varies without any clear trend. However, a feasible solution of the original problem can always be obtained based on the relaxed solution.

Next, suppose that the MCLB problem has been solved using the R mode. Then, a feasible integer solution needs to be found on the basis of it. The process is illustrated by using the sample MCLB problem given in Section 3.5 (problem instance 2 in Table 2) for α=10. Table 5 shows the module-head and module-nozzle-nozzle number assignment found by relaxation mode. The index c of variable $y^n_{l,j,c}$ means the copy number of nozzle type j on module l. Similarly, $y^h_{l,i}$ indicates the assignment of

head type i to module l. Table 6 shows the module-component assignment for the problem instance 2 in Table 2.

<Table 5 is around here>
<Table 6 is around here>

Based on Table 5, there are two copies of head type 1: one copy is assigned to module 1 and the other to module 3. There are also two copies of head type 2 in modules 2 and 4. Based on Table 1, head type 1 is compatible with nozzle types 1, 2 and 3 and head type 2 is compatible with nozzle types 4 and 5. Further, Table 5 states that the module-nozzle assignment, e.g. module 1 uses three copies of nozzle type 1 and one copy of nozzle type 3.

The assignment of Table 6 is optimal (feasible) for the relaxation mode R. The processing time of the bottleneck module 3 is 32.2×2+10×10 = 164.4. If the fractional values of the components (type1) are rounded directly: for example 7.8 → 8, 22.2 → 22, w_1= 10, then the processing times of module 1 and module 3 are 172 and 164, respectively. This means that module 1 becomes the bottleneck module. However, if the number of the components are rounded in the other way: 7.8 → 7, 22.2 → 23, w_1 = 9, then the processing times of module 1 and module 3 are 160 and 166, respectively. Module 3 becomes then the final bottleneck module. The objective value 166 is known from Table 4 to be the true optimal solution.

Next, the above trial-and-error methods were used to solve all the problem instances of Table 2 with good success. All the problems were solved optimally in this way. Although there is no guarantee that the true optimal solution can be obtained from the relaxed solution in general, a good feasible solution is found if the gap between relaxed solution and optimal solution is not large and a good feasibility restoring process is used.

Finally, the solutions by the evolutionary algorithm (EA) of Toth et al. (2009, submitted) were compared with those for the optimisation model for instances α = 0, 1 and 10. Here, the optimal solution is used to evaluate the performance of the EA heuristic method. This method divides the optimization process into three subproblems which are solved successively: assignment of heads to machine modules, assignment of component placements to machine modules and assignment of nozzles to heads. The first and the last subproblems are solved using a greedy method whereas the second one is solved by an EA. This algorithm is based on the component-to-machine mapping, similar to variable $x_{l,k}$ here in the optimization model and its objective function is (1) with the difference that the number of pick-and-place cycles w_l is relaxed to a real number. The EA of Toth et al. (2009) was applied by using 10 individuals in each population and 100 generations. The solutions of EA reported in Table 8 were the best among 20 independent runs. Accordingly, the solution times for the EA were the total times for running 20 independent runs. The solution times of EA were reported in Table 7. EA was coded in Toth et al. (2009) in MatLab and performed in a PC with 2.33 GHz (2 GB RAM) under the Windows Vista operating systems. Therefore, to facilitate comparison, an equivalent relaxation solution was also constructed from the solution of the relaxation mode (R), this approach was called R'. It is worth mentioning that the solutions obtained from approach R' coincide with the optimal solution for the related relaxation problems based on our computational experiences. Table 8 gives the GAP (%) of EA against the optimal solution. GAP measure is similar to that introduced in (32).

<Table 7 is around here>
<Table 8 is around here>

As compared with the solution time in Table 3, the solution time of EA (Table 7) is not sensitive to both problem size and parameter α. Based on Table 8, the solutions generated by EA are not bad in general. The optimality gap is reasonably small. This means that the EA has good potential to handle large size problems with different travelling factors α based on both the solution quality and solution time.

6. Conclusions

This paper investigates the machine configuration and line balancing (MCLB) problem of a printed circuit board (PCB) assembly line with modular reconfigurable machines and a single PCB type to be processed. Such lines are becoming popular due to their great flexibility. A reconfigurable machine module can be seen as a multi-head placement machine. However, the mathematical modelling of the MCLB problem has not been extensively researched so far. The MCLB problem was formulated as a non-linear integer programming model and then the non-linear model was transformed into a linear form for solution.

The MCLB problem is more complicated than the traditional versions of the line balancing problem (Baybars, 1986; Gutjahr and Nemhauser, 1964). It has to consider the component allocation and machine configuration at the same time. In addition, the machine-head assignment and machine-nozzle-component assignment were needed to solve while considering the compatibility between head and nozzle types, and the compatibility between nozzle and component types.

In the present study, this problem was solved as a two-phase process. The first phase solved the simpler problem obtained by omitting the travelling time between the feeder unit and PCB area. This resulted in a feasible machine-head assignment. In the second phase, the machine-head assignment served as an initial feasible solution for the general problem which takes also the travelling time into account. This two-phase approach improves the solution time from 2 to 5 times as compared with solving the general problem from scratch.

However, this trick does not necessarily give an optimal solution in general where the module-head assignment may be different for the first and second stage problems if the component type can be handled by more than one nozzle type, which may cause that a component type can be handled by more than one head type. A number of small-size problem instances (up to 10 component types, 5 modules, 100 component placements) were solved optimally. For a bigger problem instance, the running time with $\alpha \neq 0$ became unbearably long. A recommended action was first to solve the relaxed versions of the problem (integers are relaxed to real numbers) and then to obtain a feasible solution by adjusting the relaxed solution. For our test problems, the feasible solution obtained from the relaxed solution coincided with the optimal solution, which was obtained directly by solving the MILP formulation presented in Section 3. An advantage of the above action was much shorter running time of the solution algorithm.

Benefits of the mathematical optimisation are that one can describe the problem rigorously and get an optimal solution for at least some small-size problem instances. This can be used to evaluate the optimality performance of the heuristic approaches (see Tables 4 and 8 for this kind of comparisons). However, even the

relaxed version of the model is difficult to solve when the problem instance becomes larger.

For the future research, one can follow several directions. First, a general case of multiple assembly lines with multiple PCB-types is of practical interest. Second, the assumptions presented in Section 3 can be removed, i.e., one can then allow that a nozzle type is carried by multiple head types and a component type is handled by multiple nozzle types. Finally, it is possible to develop specialized optimisation algorithms by exploring the structure of the problem.

Acknowledgements

The first author would like to thank postdoctoral fellowship in University of Turku (Finland) and FCT (science and technology foundation) support (Portugal) through program POCI 2010 for partial funding of this research.

References

Ahmadi, R.H. and Wurgaft, H., 1994. Design for synchronized flow manufacturing. *Management Science*, 40, 1469-1483.

Amen, M., 2006. Cost-oriented assembly line balancing: Model formulations, solution difficulty, upper and lower bounds. *European Journal of Operational Research*, 168, 747-770.

Ammons, J.C., Carlyle, M., Granmer, L., Depuy, G.W., Ellis, K.P., McGinnis, L.F., Tovey, C.A. and Xu, H., 1997. Component allocation to balance workloads in printed circuit card assembly. *IIE Transactions*, 29, 265-275.

Ashayeri, J. and Selen, W., 2007. A planning and scheduling model for onsertion in printed circuit board assembly. *European Journal of Operational Research*, 183, 909-925.

Ayob, M. and Kendall, G. 2008. A survey of surface mount device placement machine optimization: machine classification. *European Journal of Operational Research*, 186, 893-914.

Ball, O.M. and Magazine, M.J., 1988. Sequencing of insertions in printed circuit board assembly. *Operations Research*, 36(2), 192-201

Baybars, I., 1986. A survey of exact algorithms for the simple assembly line balancing problem. *Management Science*, 32, 909-932.

Becker, C. and Scholl, A., 2006. A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, 168, 694-715.

Choudhury, N.D., Wilhelm, W.E., Rao, B., Gott, J. and Khotekar, N., 2007. Process planning for circuit card assembly on a series of dual head placement machines. *European Journal of Operational Research*, 182, 626-639.

Grunow, M., Gunther, H.O., Schleusener, M. and Yilmaz, I.O., 2004. Operations planning for collect-and-place machines in PCB assembly. *Computer & Industrial Engineering*, 47, 409-429.

Gutjahr, A.L. and Nemhauser, G.L., 1964. An algorithm for the line balancing problem. *Management Science*, 11, 308-315.

Häyrinen, T., Johsson, M., Johtela, T., Smed J. and Nevalainen, O., 2000. Scheduling algorithms for computer-aided line balancing in printed circuit board assembly. *Production Planning and Control*, 11, 497-510.

Hiller, M.S. and Brandeau, M.L., 2001. Cost minimization and workload balancing in printed circuit board assembly. *IIE Transactions*, 33, 547-557.

Ho, W., Ji, P. and Dey, P.K., 2008. Optimization of PCB component placements for the collect-and-place machines. *International Journal of Advanced Manufacturing Technology*, 37, 828-836.

Ji, P., Sze, M.T. and Lee, W.B., 2001. A genetic algorithm of determining cycle time for printed circuit board assembly. *European Journal of Operational Research*, 128, 175-184.

Khoo, L.P. and Alisantoso, D., 2003. Line balancing of PCB assembly line using immune algorithms. *Engineering with Computers*, 19, 92-100.

Kodek, D.M. and Krisper, M., 2004. Optimal algorithm for minimizing production cycle time of a printed circuit board assembly line. *International Journal of Production Research*, 42(23), 5031-5048.

Kulak, O., Yilmaz, I.O. and Gunther, H.O., 2008. GA-based solution approach for balancing printed circuit board assembly line. *OR Spectrum*, 30, 469-491.

Lapierre, S.D, Debargis L. and Soumis F., 2000. Balancing printed board assembly line systems. *International Journal of Production Research*, 38(16), 3899-3911.

Lapierre, S.D., Ruiz, A. and Soriano, P., 2006. Balancing assembly lines with tabu search. *European Journal of Operational Research*, 168, 826-837.

Leipälä, T. and Nevalainen, O., 1989. Optimization of the movements of a component placement machine. *European Journal of Operational Research*, 38, 167-177.

Lin, W.L. and Tardif, V., 1999. Component partitioning under demand and capacity uncertainty in printed circuit board assembly. *International Journal of Flexible Manufacturing Systems*, 11, 159-176.

Scholl, A. and Becker, C., 2006. State-of-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, 168, 666-693.

Sze, M.T., Ji, P. and Lee, W.B., 2001. Modeling component assignment problem in PCB assembly. *Assembly Automation*, 21, 55-60.

Tazari, S., Muller-Hannemann, M. and Weihe, K., 2006. Workload balance in multi-stage production processes. *Lecture Notes in Computer Science*, 4007, 49-60.

Tirpak, T.M., Mohapatra, P.K., Nelson, P.C. and Rajbhandari, R.R., 2002. A generic classification and object-oriented simulation toolkit for SMT assembly equipment. IEEE *Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, 32(1), 104-121.

Toth, A., Knuutila, T. and Nevalainen, O.S., 2009. Reconfiguring flexible machine modules of a PCB assembly line (in submission).

Wan, Y.F. and Ji P., 2001. A tabu search heuristic for the component assignment problem in PCB assembly. *Assembly Automation*, 21(3), 236-240.

Yildirim, MB., Duman, E. and Duman, D., 2006. Dispatching rules for allocation of component types in automated assembly of printed circuit boards. *Lecture Notes in Computer Science*, 4263, 55-64.

Yilmaz, I.O., Gunther H.O. and Jain S., 2009. Simulation of mixed model PCB assembly lines with group setup and bypass conveyors. *International Journal of Advanced Manufacturing Technology*, 42(3-4), 335-347.

Figure 1. Modular component placement systems.

Table 1. A sample MCLB problem.

Table 2. Three small-size problem instances and one big problem instance (instance 4) and related sizes of the mixed linear integer programming model (number of constraints ×number of decision variables).

Table 3. Solution times (seconds) of the optimisation model for the problem instances using optimisation mode (O), and relaxation mode (R). O/R is the ratio of solution time for the optimisation mode against relaxation mode. For instance 4 with α >1, it was too time-consuming to solve the problem. The tests were performed in a 2.79 GHz(1 GB RAM) Pentium 4 PC under the Windows XP operating systems.

Table 4. Values of the objective functions for the optimisation mode (O) and relaxation mode (R) as well as the gap of the relaxed solution. For instance 4 with α >1, it was too time-consuming to solve the problem.

Table 5. Module-head and module-nozzle-nozzle copy assignment for the problem instance 2 in Table 2.

Table 6. Module-component assignment for the relaxation mode for the problem instance 2 in Table 2.

Table 7. Solution times (seconds) of EA for running 20 independent runs. The EA was coded in Matlab and performed in a PC with 2.33 GHz (2 GB RAM) under the Windows Vista operating systems.

Table 8. Comparison of EA solutions against optimal solution.

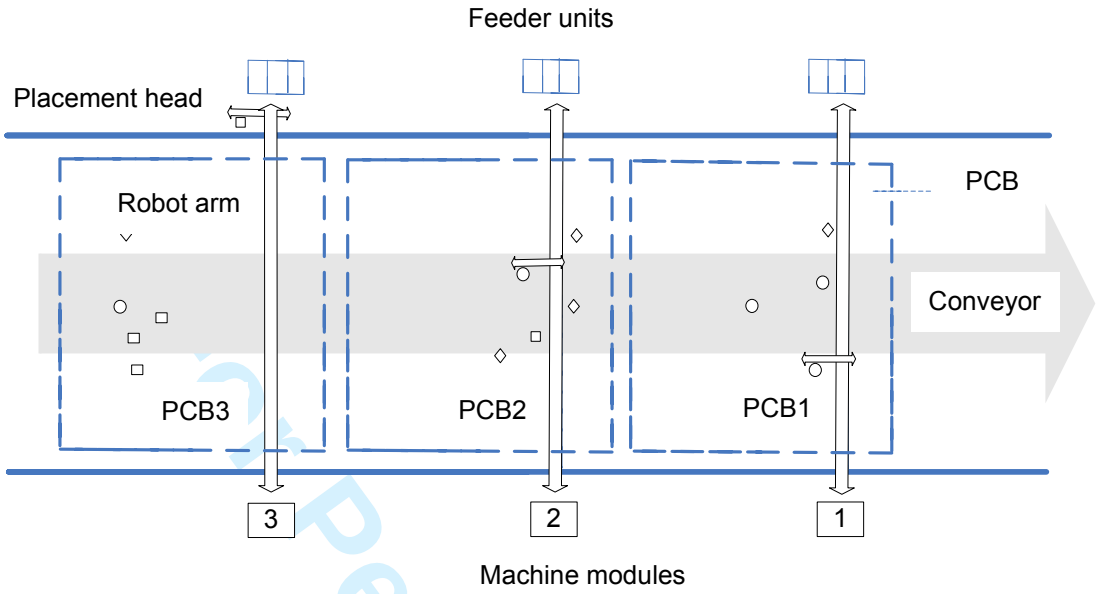

Figure 1. Modular component placement systems.

Table 1. A sample MCLB problem.

Number of component types |**A**|		8
Number of machine modules |**L**|		4
Number of head types |**H**|		2
Number of nozzle types |**N**|		5
Number of placements for component type k, D_k ($k \in \mathbf{A}$)		$D_1 = 30$; $D_2 = 20$; $D_3 = 10$; $D_4 = 5$; $D_5 = 3$; $D_6 = 3$; $D_7 = 2$; $D_8 = 1$;
Head-nozzle compatibility matrix B^{hn} (|**H**|×|**N**|=2×5)		$\begin{vmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{vmatrix}$
Component-nozzle compatibility matrix B^{an} (|**A**|×|**N**|= 8×5)		$\begin{vmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{vmatrix}$
Feeder width of component type k, E_k ($k \in \mathbf{A}$)		$E_1 = 1$; $E_2 = 2$; $E_3 = 3$; $E_4 = 4$; $E_5 = 5$; $E_6 = 5$; $E_7 = 5$; $E_8 = 7$;
Number of nozzles attached to head type i, G_i ($i \in \mathbf{H}$)		$G_1 = 1$; $G_2 = 2$
Feeder capacity F_l ($l \in \mathbf{L}$)		$F_1 = F_2 = F_3 = F_4 = 12$
Time factors	Picking and placing time, T_l^{pp} ($l \in \mathbf{L}$)	$T_1^{pp} = T_2^{pp} = T_3^{pp} = T_4^{pp} = 2$
	Travelling time T_l^{tr} ($l \in \mathbf{L}$)	$T_1^{tr} = T_2^{tr} = T_3^{tr} = T_4^{tr} = 1$

Table 2. Three small-size problem instances and one big problem instance (instance 4) and related sizes of the mixed linear integer programming model (number of constraints ×number of variables).

Instance	Component types $\|\mathbf{A}\|$	Modules $\|\mathbf{L}\|$	Head types $\|\mathbf{H}\|$	Nozzle types $\|\mathbf{N}\|$	Placements $\Sigma_k\|D_k\|$	Feeder capacity F	Variables		Constraints	
							$\alpha=0$	$\alpha\neq0$	$\alpha=0$	$\alpha\neq0$
1	8	5	2	5	55	10	121	166	131	248
2	8	4	2	5	74	12	97	131	108	200
3	10	5	3	8	100	20	161	331	170	550
4	20	8	4	10	320	30	441	753	452	1156

Table 3. Solution times (seconds) of the optimisation model for the problem instances using optimisation mode (O), and relaxation mode (R). O/R is the ratio of solution time for the optimisation mode against relaxation mode. For instance 4 with α >1, it was too time-consuming to solve the problem. The tests were performed in a 2.79 GHz(1 GB RAM) Pentium 4 PC under the Windows XP operating systems.

Instance	$\alpha = 0$	$\alpha = 1$			$\alpha = 2$			$\alpha = 5$			$\alpha = 10$		
	O(R)	O	R	O/R	O	R	O/R	O	R	O/R	O	R	O/R
1	0.31	150	78	1.9	832	72.7	11.4	704	79.5	8.9	3215	171	18.8
2	0.2	6.7	4.9	1.4	8.2	4.8	1.7	51.7	3.2	16.2	93	7.9	11.8
3	0.24	2684	1437	1.9	1299	806	1.6	22733	660	34.4	15339	391	39.2
4	2.2	51208	419	122	--	--	--	--	--	--	--	--	--

Table 4. Solutions of the optimisation mode (O) and relaxation mode (R) as well as the gap of the relaxed solution. For instance 4 with α >1, it was too time-consuming to solve the problem.

Instance	$\alpha = 0$	$\alpha = 1$			$\alpha = 2$			$\alpha = 5$			$\alpha = 10$		
	O(R)	O	R	GAP(%)	O	R	GAP(%)	O	R	GAP(%)	O	R	GAP(%)
1	34	39	37.8	3.08	44	42.2	4.09	59	55.6	5.76	84	78	7.38
2	68	78	77.4	0.77	88	86.9	1.25	116	116	0.34	166	164	0.96
3	90	96	95.8	0.21	102	102	0.29	120	119	0.67	150	149	0.93
4	112	120	119	0.75	--	--	--	--	--	--	--	--	--

Table 5. Module-head and module-nozzle-nozzle copy assignment for the problem instance 2 in Table 2.

Module-head	$y^h_{1,1}=1, y^h_{2,2}=1, y^h_{3,1}=1, y^h_{4,2}=1$
Module-nozzle-nozzle copy	$y^n_{1,1,1}=y^n_{1,1,2}=y^n_{1,1,3}=1, y^h_{1,3,1}=1$
	$y^n_{2,4,1}=1$
	$y^n_{3,1,1}=y^n_{3,1,2}=y^n_{3,1,3}=1, y^n_{3,2,1}=1$
	$y^n_{4,5,1}=1$

Table 6. Module-component assignment for the relaxation mode for the problem instance 2 in Table 2.

Module \ Component	1	2	3	4	5	6	7	8	w_l
1	7.8	20		5	3				9.27
2						3			3
3	22.2		10						10
4							2	1	3
D_k	30	20	10	5	3	3	2	1	

Table 7. Solution times (seconds) of EA for running 20 independent runs.

Instance	$\alpha = 0$	$\alpha = 1$	$\alpha = 10$
1	354	350	344
2	306	307	308
3	474	480	480
4	500	510	510

Table 8. Comparison of EA solutions against optimal solution.

Instance	α = 0			α = 1			α = 10		
	O(R')	EA	GAP(%)	R'	EA	GAP(%)	R'	EA	GAP(%)
1	34	34	0	38.3	38.3	0	78	78	0
2	68	68	0	78	78	0	166	168	1.2
3	90	90	0	96	96.7	0.73	149.6	152	1.6
4	112	112	0	120	121	0.83	--	190	--