2

Optimization Techniques

2.1 Introduction

There are mainly two approaches to optimizing PCB assembly problems. The first one is to formulate the problems as mathematical models and then solve them to optimality using exact algorithms or commercial packages. The second one is simply to generate good solutions of the problems using metaheuristics. Although the latter approach can generate solutions efficiently, no one knows how good the solutions are unless the optimal solution is known in advance.

This chapter is organized in the following way: Section 2.2 presents several types of mathematical programming in Operations Research, which can be applied to PCB assembly problems. Section 2.3 and Section 2.4 survey commonly used exact algorithms and metaheuristics, respectively. Section 2.5 describes the commercial packages including those adopted in this book and some other prevalently used ones. Finally, some remarks concerning this chapter are summarized in Section 2.6.

2.2 Mathematical Programming

Many researchers used mathematical programming models in dealing with the PCB assembly problems. A mathematical programming model is a mathematical representation of the actual situation that may be used to make better decisions or simply to understand the actual situation better (Winston and Venkataramanan, 2003). The common feature which mathematical programming models have is that they all involve optimization (Williams, 1999). In PCB assembly problems, optimization includes the minimization of something (*e.g.*, setup time, placement time, and so on) or the maximization of something (*e.g.*, throughput, workload balance, and so on), under certain constraints (*e.g.*, machine capacity, available production time, and so on).

In the following subsections, attention is confined to linear programming models, integer linear programming models, and nonlinear programming models.

They are presented and studied because the component sequencing and the feeder arrangement problems for the sequential pick-and-place machine (in Chapter 3) and the concurrent chip shooter machine (in Chapter 4), and the line assignment and the component allocation problems (in Chapter 5) can be formulated with these types of models.

2.2.1 Linear Programming

A model is defined as linear program (LP) when the objective function and the constraints involve linear expressions and the decision variables are continuous. Comparatively, LP models are given so much attention in comparison with nonlinear programming models because they are much easier to solve. The transportation model, first described by Hitchcock in 1941, is a special class of LP (Williams, 1999). Suppose that a number of suppliers (i = 1, 2, …, m) provides a commodity to a number of customers (j = 1, 2, …, n). The transportation problem is how to meet each customer's requirement, d_j, while not exceeding the capacity of any supplier, s_i, at minimum cost, c_{ij}. By introducing variables x_{ij} to represent the quantity of the commodity sent from supplier i to customer j, the transportation model can be written as (Winston and Venkataramanan, 2003)

$$\text{Minimize } z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{2.1}$$

subject to

$$\sum_{j=1}^{n} x_{ij} \le s_i \qquad i = 1, 2, \ldots, m \tag{2.2}$$

$$\sum_{i=1}^{m} x_{ij} \ge d_j \qquad j = 1, 2, \ldots, n \tag{2.3}$$

$$\text{All } x_{ij} \ge 0. \tag{M2-1}$$

The objective function (2.1) is to minimize the total transportation cost. Constraint set (2.2) is known as a supply or availability constraint, whereas constraint set (2.3) is known as a demand or requirement constraint. M2-1 is referred to as the transportation model. If the total supply equals total demand, then the problem is said to be a balanced transportation problem. In this case, constraint sets (2.2), and (2.3) are treated as both "=" instead of "≤" and "≥", respectively.

2.2.2 Integer Linear Programming

Integer linear programming or integer programming (IP) is widely adopted as a method of modeling because some variables are not continuous but integers in many cases in real life. Actually, IP is a subset of LP, with an additional constraint that some or all decision variables are restricted to integral values depending on the

type of IP. Generally, there are three types of IP. First, IP is called pure integer linear programming if all variables must be integers. Second, IP is called mixed integer linear programming if only some of the variables must be integers. Third, IP is called binary integer linear programming if all the variables must be either 0 or 1 (Winston and Venkataramanan, 2003).

IP has important practical applications. However, it was pointed out that computational experience with IP has been less than satisfactory (Taha, 2003).

The traveling salesman problem (TSP) is one of the most widely studied IP problems. The TSP can be easily stated as follows. A salesman wants to visit n distinct cities and then return home. He wants to determine the sequence of the travel so that the overall travel distance is minimized while visiting each city not more than once. Although the TSP is conceptually simple, it is difficult to obtain an optimal solution. In an n-city situation, any permutation of n cities yields a possible solution. As a consequence, $n!$ possible tours must be evaluated in the search space. By introducing variables x_{ij} to represent the tour of the salesman travels from city i to city j, one of the common IP formulations for the TSP can be written as (Winston and Venkataramanan, 2003)

$$\text{Minimize } z = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} c_{ij} x_{ij} \tag{2.4}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, 2, \ldots, n; i \neq j \tag{2.5}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, 2, \ldots, n; i \neq j \tag{2.6}$$

$$u_i - u_j + n x_{ij} \leq n - 1 \qquad i, j = 2, 3, \ldots, n; i \neq j \tag{2.7}$$

$$\text{All } x_{ij} = 0 \text{ or } 1. \text{ All } u_i \geq 0 \text{ and is a set of integers.} \tag{M2-2}$$

The distance between city i and city j is denoted as c_{ij}. The objective function (2.4) is simply to minimize the total distance traveled in a tour. Constraint set (2.5) ensures that the salesman arrives once at each city. Constraint set (2.6) ensures that the salesman leaves each city once. Constraint set (2.7) is to avoid the presence of a subtour.

The TSP formulated for the component sequencing problem is known as the Euclidean TSP, in which the distance matrix c is expected to be symmetrical, that is, $c_{ij} = c_{ji}$ for all i, j, and to satisfy the triangle inequality, that is, $c_{ik} \leq c_{ij} + c_{jk}$ for all distinct i, j, k.

2.2.3 Nonlinear Programming

In the previous subsections, LP as well as IP has been studied. For an LP, the objective is to minimize or maximize a linear function subject to linear constraints. Although LP problems are very common and cover a wide range of problems, the objective function may not be a linear function, or some of the constraints may not be linear in a real-life situation. Such an optimization problem is called a nonlinear programming (NLP) problem.

The quadratic assignment problem (QAP) is a generalization of the linear assignment problem. The major difference between them is that the objective function of the QAP is in a nonlinear expression. Therefore, it is comparatively difficult to solve. The QAP can be described as follows. Consider a set of facilities (i, k = 1, 2, …, n) placed uniquely in a set of locations (j, l = 1, 2, …, n). The workflow intensity between each pair of facilities is a_{ik} while the distance between each pair of locations is b_{jl}. Also, a fixed cost c_{ij} associated with the placement of facility i in location j is specified. The formulation of the QAP can be written as (Burkard *et al.*, 1991; Williams, 1999)

$$\text{Minimize } z = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{\substack{k=1\\k\neq i}}^{n}\sum_{\substack{l=1\\l\neq j}}^{n} a_{ik}b_{jl}x_{ij}x_{kl} + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{2.8}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, 2, \ldots, n \tag{2.9}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, 2, \ldots, n \tag{2.10}$$

$$\text{All } x_{ij} = 0 \text{ or } 1. \tag{M2-3}$$

The decision variables x_{ij} represent the placement of facility i in location j. Often, the objective function (2.8) is to assign facilities to locations so that the travel distance of material flow is minimized, while assuming that the cost of assigning a facility does not depend upon the location, that is, $c_{ij} = 0$. Constraint set (2.9) ensures that each location must be occupied by only one facility. Constraint set (2.10) ensures that each facility must be assigned only to one location.

2.3 Exact Algorithms

A set of fixed computational rules for solving a particular class of problems or models is known as an algorithm. It applies the rules repetitively to the problem or the model, each iteration moves the solution closer to the optimum. In Operations Research, there does not exist an algorithm that solves all types of mathematical models. For example, the simplex algorithm is the general method for solving LP

models, whereas the branch-and-bound algorithm is the general technique for solving IP models.

2.3.1 Algorithms for Linear Programming

Finding an optimal solution to an LP model can be regarded as assigning values to the decision variables so that the specified objective is achieved and the constraints are not violated. In the following, two commonly used algorithms for solving the LP models are discussed: the simplex algorithm and the interior point algorithm.

2.3.1.1 The Simplex Algorithm

The simplex algorithm has proved highly efficient in practice and therefore was widely adopted in commercial optimization packages for solving any LP model (Jensen and Bard, 2003). Its development was based on the graphical method that the optimal solution is always associated with a corner point of the solution space. The idea of the simplex algorithm is to move the solution to a new corner that has the potential to improve the value of the objective function in each iteration. The process terminates when the optimal solution is found (Taha, 2003).

2.3.1.2 The Interior Point Algorithm

The simplex algorithm searches for the optimal solution along the corner points of the solution space, whereas the interior point algorithm looks for the optimum through the interior of the feasible region (Jensen and Bard, 2003). The interior point algorithm has theoretical importance that it provides a bound on the computational effort required to solve a problem that is a polynomial function of its size. But, there is no polynomial bound available in the simplex algorithm (Carter and Price, 2001; Jensen and Bard, 2003).

2.3.2 Algorithms for Integer Linear Programming

Unlike LP with the simplex algorithm, a good IP algorithm for a very wide class of IP problems has not been developed (Williams, 1999). Different algorithms are good with different types of problem. Generally, IP algorithms are based on exploiting the tremendous computational success of LP. Thus, before applying an IP algorithm, the integer restriction on the problem should be relaxed first to form an LP model. Starting from the continuous optimum point obtained from the LP model, integer constraints are incorporated repeatedly to modify the LP solution space in a manner that will eventually render the optimum extreme point satisfying the integer requirements.

2.3.2.1 The Branch-and-Bound Algorithm

In practice, the branch-and-bound (B&B) algorithm is widely used for solving IP models, especially mixed integer linear programming (MIP) models (Williams, 1999). The idea of the B&B algorithm is to perform the enumeration efficiently so that not all combinations of decision variables must be examined. Sometimes, the terms “implicit enumeration”, “tree search”, and “strategic partitioning” are used depending on the implementation of the algorithm (Jensen and Bard, 2003).

The B&B algorithm starts with solving an IP model as an LP model by relaxing the integrality conditions. In case the resultant LP solution or the continuous optimum is an integer, this solution will also be the integer optimum. Otherwise, the B&B algorithm sets up lower and upper bounds for the optimal solution. The branching strategy repetitively decreases the upper bound and increases the lower bound. The process terminates, provided that the processing list is empty (Castillo *et al.*, 2002).

2.3.2.2 The Cutting Plane Algorithm

As with the B&B algorithm for solving IP models, the cutting plane algorithm relaxes the integrality requirements of the IP models and solves the resultant LP. But rather than repetitively imposing restrictions on the fractional variables, as is done in the B&B algorithm, extra constraints (*i.e.*, cutting planes) are systematically added to the model, and the model is then resolved. The new solution to the further constrained model may or may not be an integer. By continuing the process until an integer solution is found or the model is shown to be infeasible, the IP model can be solved (Williams, 1999; Jensen and Bard, 2003).

2.3.3 Algorithms for Nonlinear Programming

The quadratic assignment problem (QAP), as shown in Section 2.2.3, belongs to the NLP model because there is a nonlinear expression in the objective function. In addition, the QAP is a binary NLP model as the decision variable is either 0 or 1. But, if an NLP model consists of both integer and continuous variables, it is regarded as mixed integer nonlinear programming model (MINLP). In this book, the integrated problem for the concurrent chip shooter machine will be formulated as this type of model. Therefore, the algorithms for solving the MINLP models are discussed.

Actually, the MINLP problems are the most difficult optimization problems of all. They combine all the difficulties of both the MIP as well as the NLP. Also, they do not have the properties of the MIP or the NLP. For example, a local minimum is equivalent to the global minimum for convex NLP problems. But this result does not hold for MINLP problems. Therefore, MINLP problems belong to the class of NP-complete problems (Kallrath, 1999).

There are two categories of MINLP problems: convex and nonconvex. In the following, the generalized benders decomposition, an algorithm for solving convex MINLP problems, and the branch-and-reduce algorithm, an algorithm for optimizing nonconvex MINLP problems, are described in brief.

2.3.3.1 The Generalized Benders Algorithm

In the generalized benders decomposition, two sequences of updated upper and lower bounds are generated. The upper bounds correspond to solving subproblems in continuous variables by fixing the integer variables, while the lower bounds are based on duality theory. The algorithm terminates if the lower and the upper bounds equal or cross each other (Floudas, 2000).

2.3.3.2 The Branch-and-Reduce Algorithm

The branch-and-reduce algorithm is the extended version of the B&B algorithm for optimizing nonconvex MINLP problems in which valid convex underestimating NLPs can be constructed for the nonconvex relaxation. Due to the fact that nonconvex NLPs must be underestimated at each node, convergence can be achieved only if the continuous variables are branched. A number of tests are suggested to speed up the reduction of the solution space, including the optimality-based range reduction tests and the feasibility-based range reduction tests (Floudas, 2000; Tawarmalani and Sahinidis, 2002).

2.4 Metaheuristics

For many exact algorithms, the computational effort required is an exponential function of the problem size. In a sense, therefore, it may be necessary to abandon the search for the optimal solution using the exact algorithms and simply seek a good solution in a reasonable computational time using heuristics. In Operations Research, the term "heuristic" refers to the methods for the problem under study, based on rule of thumb, common sense, or adaptations of exact methods for simpler models. They are used to find reasonable solutions when the problems are complex and difficult to solve. In optimization, a heuristic method refers to a practical and quick method based on strategies that are likely to (but not guaranteed to) lead to a solution that is approximately optimal or near-optimal (Murty, 1995).

Heuristics can be classified as either constructive (greedy) heuristics or as local search heuristics (Walser, 1999). First, greedy heuristics, such as the nearest neighbor heuristic, simply list all the feasible solutions of the problem under study, evaluate their objective functions, and pick the best as the output of the model. This approach of complete enumeration is likely to be grossly inefficient especially when the number of possible solutions to the problem is vast. So, greedy heuristics are not desirable for solving combinatorial optimization problems, and conversely, local search heuristics are more suitable.

Second, local search heuristics, such as the 2-opt local search heuristic, are based on the concept of exploring the vicinity of the current solution. Neighboring solutions are generated by a move-generation mechanism. If the generated neighbor has a better objective value, it becomes a new current solution, or otherwise the current solution is retained. The process is iterated until there is no possibility of improvement in the neighboring solution. The method then terminates at a point called local optimum, which may be far from any global optimum, as shown in Figure 2.1. This is one of the disadvantages of simple local search methods.

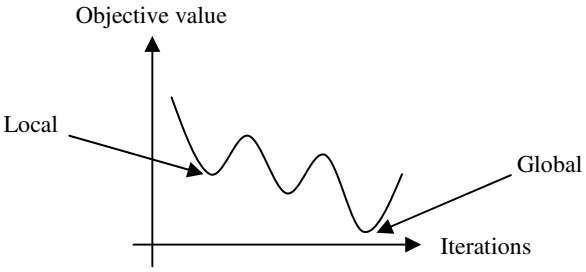

Figure 2.1. Global and local optimum

To avoid getting trapped at a local optimum, a number of conceptual metalevel strategies have been developed for local search heuristics. These strategies are referred to as metaheuristics (Osman and Kelly, 1996). A metaheuristic is an iterative generation process that guides a subordinate or simple heuristic by combining intelligence, biological evolution, neural systems, and statistical mechanics for exploring and exploiting the search spaces using learning strategies to structure information to find near-optimal solutions efficiently. The families of metaheuristics include genetic algorithms, the greedy random adaptive search procedure, problem-space search, neural networks, simulated annealing, tabu search, threshold algorithms, and their hybrids.

In the following subsections, three metaheuristics will be described briefly. These three approaches are very general and applicable to a wide range of problems, while yielding reasonable performance in terms of speed and good performance in terms of the quality of the solutions generated. In addition, the Committee on the Next Decade of Operations Research has singled out these approaches as "extremely promising" for the future treatment of practical applications (Glover *et al.*, 1993). They are simulated annealing (SA), tabu search (TS), and genetic algorithms (GAs).

2.4.1 Simulated Annealing

Simulated annealing (SA), introduced by Kirkpatrick, Gelatt, and Vecchi in 1983, is a technique combining the concepts of statistical mechanics. SA is based on an analogy between the annealing process and the technique of solving the combinatorial optimization problem. SA starts with an initial solution and repeatedly generates a neighbor solution. A neighbor solution is always accepted if there is an improvement in the objective value. However, if it is worse, the solution may be accepted and this acceptance will depend on the control parameter (temperature). To apply SA to a practical problem, there are several choices to be made. The choices can be divided into two main categories: problem-specific and generic. Table 2.1 illustrates the several choices (Johnson *et al.*, 1989; Rayward *et al.*, 1993; Osman and Kelly, 1996).

Table 2.1. Choices to be made in implementing simulated annealing

Problem-specific:	- What is the solution representation?
	- What is the objective function?
	- What are the neighborhood generation mechanisms?
	- How do we determine an initial solution?
Generic:	- How do we determine an initial temperature?
	- What is the temperature update rule?
	- How many iterations must be performed at each temperature?
	- What is the stopping criterion?

2.4.2 Tabu Search

Tabu search (TS) was developed by Glover and Hansen in 1986 for solving combinatorial optimization problems. TS, like SA, is based on local search heuristics with local-optima avoidance, but in a deterministic way which tries to model human memory processes. In other words, TS is an iterative metaheuristic search procedure combining the concepts of artificial intelligence.

TS starts with an initial current solution, which can be generated randomly. Then, the method generates a list of all neighborhood solutions, which is known as the candidate list, from the current solution. Next, all solutions in the candidate list are evaluated, and the best solution from the candidate list will be selected. Sometimes, the best solution may not be selected if the solution is in the tabu memory lists. The lists can be divided into two parts, which are recency (short-term) memory, and frequency (long-term) memory. Both memories are responsible for recording the history of the search, and especially the moves, called attributes, have participated in generating past solutions. The mechanism attempts to avoid the cycling behavior of the method. If the selection is forbidden (tabu), the method proceeds to select the second best solution in the candidate list as the new current solution. On the other hand, if the current solution is better than the specified aspiration level or best fitness value found so far, the solution's tabu status is overridden and the solution is still admissible as the next current solution. The current best solution is updated if necessary, and then a new list of candidate solutions is generated around the new current solution. The procedure continues until the stopping criteria are satisfied (Glover *et al.*, 1993; Glover and Laguna, 1997).

2.4.3 Genetic Algorithms

Genetic algorithms (GAs) were developed by Holland in the 1960s. Only recently, their potential for solving combinatorial optimization problems has been explored. Similar to SA and TS, GAs can avoid getting trapped in a local optimum by the aid of one of the genetic operators called mutation. Actually, the basic idea of GAs is to maintain a population of candidate solutions that evolves under selective

pressure. Hence, they can be viewed as a class of local search based on a solution-generation mechanism operating on attributes of a set of solutions rather than attributes of a single solution by the move-generation mechanism of the local search methods, like SA and TS (Osman and Kelly, 1996). As GA is selected as a heuristic method to solve the problems in this book, it will be described more thoroughly in the following chapters.

In recent years, many researchers discovered that a simple GA was not desirable for solving combinatorial optimization problems with a large problem size. Therefore, they incorporated local search heuristics into the GA, which is called the genetic local search (GLS), for solving the TSP (Freisleben and Merz, 1996a,b) and the QAP (Huntley and Brown, 1996; Ahuja *et al.*, 2000). Experimental results showed that the GLS could solve the TSP and the QAP effectively. For instance, it was found that the GLS obtained the optimal solution for a TSP with 1,400 cities after 200 iterations (Freisleben and Merz, 1996b).

Commonly used local search heuristics can be classified into three main categories: 2-opt, 3-opt, and Lin-Kernighan (LK). For the 2-opt local search heuristic, a neighboring solution is obtained from the current solution by deleting two edges, reversing and reconnecting the two resultant paths in a different way to form a new tour. For the 3-opt local search heuristic, three edges are deleted instead of two. The resultant paths are combined in the best possible way. 3-opt is much more effective than 2-opt, though the size of the neighborhood is larger, and hence more time-consuming to search. To improve 3-opt further, Lin and Kernighan developed a sophisticated edge exchange procedure where the number of edges to be exchanged is variable (Reinelt, 1994).

2.5 Commercial Packages

It is worth writing very sophisticated and efficient computer programs for algorithms when they are used frequently for solving many different models. Such programs, usually consisting of a number of algorithms collected together, are called a “package” of computer routines. Many such package programs are available commercially for solving mathematical programming models. When a mathematical programming model is built, it is usually worth using an existing package to solve it rather than getting diverted onto the task of programming the computer to solve the model oneself (Williams, 1999).

In this book, integrated problems for the sequential pick-and-place machine and the concurrent chip shooter machine are formulated as integer nonlinear programming models presented in Chapter 3 and Chapter 4, respectively. Then, each of the models is equivalently converted into an integer linear programming model. To verify the models, two commercial packages are used. First, BARON is adopted to solve integer nonlinear programming models, whereas CPLEX is used for optimizing integer linear programming models. In the following, the characteristics and the working principles of both commercial packages are described briefly. In addition, some existing commercial packages not used in this book are discussed.

2.5.1 BARON

BARON is a computational system for solving nonconvex optimization models to global optimality. Purely continuous NLP models, purely integer NLP models, and MINLP models can be solved with the software. This is the reason why it is adopted to solve the integer nonlinear programming models presented in Chapter 3 and Chapter 4. BARON combines constraint propagation, interval analysis, and duality for efficient range reduction with rigorous relaxation constructed by enlarging the feasible region and/or underestimating the objective function.

2.5.2 CPLEX

CPLEX is used as an integer linear programming solver in this book because it is powerful in solving LP and MIP problems. For problems with integer variables, CPLEX uses a branch-and-bound search with modern algorithmic features, such as cuts and heuristics, to solve a series of LP subproblems. Because a single MIP generates many LP subproblems, even a small MIP can be very computationally intensive and requires significant amounts of physical memory.

2.5.3 Others

Nowadays, there are numerous commercial packages available for tackling different types of mathematical programming problems. For instance, apart from BARON, DICOPT is a framework for solving MINLP models using standard MIP and NLP solvers to solve MIP and NLP subproblems generated by the algorithm. SBB is another MINLP solver. It is based on a combination of the standard B&B algorithm and some of the standard NLP solvers for subproblems.

On the other hand, except for CPLEX, LINDO is also widely used as an MIP solver. The base version includes primal and dual simplex solvers. For models with integer restrictions, LINDO includes an exceptional integer solver with default settings selected to work well on broad classes of integer models. OSL includes a set of stand-alone solvers for the MIP problem. A branch-and-bound technique is used for MIP, whereas the simplex algorithm is used to solve LP subproblems.

2.6 Summary

Various optimization techniques appropriate for the PCB assembly problems have been discussed in this chapter. Some remarks concerning these techniques are summarized as follows:

1. PCB assembly problems can be formulated as different types of mathematical programming models, including linear programming, integer linear programming, and nonlinear programming models.
2. An algorithm does not exist that solves all types of mathematical models. For instance, the simplex algorithm is the general method for solving linear programming models but not integer linear programming models.

3. Simulated annealing, tabu search, and the genetic algorithms (GAs) are commonly used metaheuristics. Note that each metaheuristic possesses its own characteristics and there is no special one that is acknowledged as the best.
4. GAs will be adopted to solve integrated problems for both types of placement machines, the line assignment problem, and the component allocation problem. The reason is that GAs have been applied successfully in a wide variety of optimization problems such as the TSP, the QAP, and the minimum spanning tree problem (Gen and Cheng, 1997). In addition, the merits of GAs, including simplicity, ease of operation, and flexibility, are the encouraging factors for applying it.
5. BARON and CPLEX are commercial packages used to solve integer nonlinear programming models and integer linear programming models to be formulated in this book, respectively.

In the next chapter, the component sequencing problem and the feeder arrangement problem are studied for the sequential pick-and-place machine. Each of the problems is formulated as an individual mathematical model first. Because of their inseparable relationship, one cannot be solved unless the solution of the other one is obtained beforehand. Therefore, two mathematical models in nonlinear form are constructed for the integrated problem. The nonlinear programming models are also converted equivalently into two linear programming models. These models are compared in terms of computing complexity as well as the computational time taken for obtaining the global optimal solution. To achieve this goal, BARON and CPLEX are used. Besides applying mathematical modeling, a genetic algorithm incorporating several improved heuristics is developed.

http://www.springer.com/978-1-84628-499-1

Optimal Production Planning for PCB Assembly

Ho, W.; Ji, P.

2007, IX, 121 p., Hardcover
ISBN: 978-1-84628-499-1