Computers & Industrial Engineering 64 (2013) 1084–1092

Contents lists available at SciVerse ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

# An integrated algorithm for cutting stock problems in the thin-film transistor liquid crystal display industry

Hao-Chun Lu [a], Yao-Huei Huang [b,*], Kuo-An Tseng [c]

[a] Department of Information Management, Fu Jen Catholic University, No. 510, Zhongzheng Rd., Xinzhuang Dist., New Taipei City 24205, Taiwan
[b] Institute of Information Management, National Chiao Tung University, Management Building 2, 1001 Ta-Hsueh Rd., Hsinchu 300, Taiwan
[c] Department of Finance, Lunghwa University of Science and Technology, 300, Sec. 1, Wanshou Rd., Guishan, Taoyuan 33306, Taiwan

A R T I C L E I N F O

*Article history:*
Received 14 September 2012
Received in revised form 21 December 2012
Accepted 7 February 2013
Available online 16 February 2013

*Keywords:*
Cutting stock problem
TFT-LCD
MIP
Genetic algorithm
Corner arrangement
Production plan

A B S T R A C T

The cutting stock problem (CSP) is a critical issue in the manufacturing of thin film transistor liquid crystal display (TFT-LCD) products. Two manufacturing processes are utilized in this industry: (1) various TFT-LCD plates are cut from a glass substrate based on cutting patterns, and (2) the number of glass substrates required to satisfy customer requirements is minimized. The current algorithm used to select the cutting pattern is defined as a mixed integer program (MIP). Although the current MIP method yields an optimal solution, but the computation time is unacceptable when the problem scale is large. To accelerate the computation and improve the current method, this study proposes an integrated algorithm that incorporates a genetic algorithm, a corner arrangement method, and a production plan model to solve CSPs in the TFT-LCD industry. The results of numerical experiments demonstrate that the proposed algorithm is significantly more efficient than the current method, especially when applied to large-scale problems.



## 1. Introduction

With the rapid growth and spread of information technology, the cutting stock problem (CSP) has become a critical issue in the manufacturing of thin-film transistor liquid crystal display (TFT-LCD) products. Increasing attention is being directed toward cutting issues in various manufacturing industries (e.g., textiles, leather, paper, wood, glass, and sheet metal). Two manufacturing processes are used in the TFT-LCD industry: (1) various TFT-LCD plates are cut from a glass substrate based on cutting patterns, and (2) the number of glass substrates required to satisfy customer requirements is minimized. However, the ideal cutting patterns are exceedingly complicated. To determine the effectiveness and complexity of the search procedure, the CSP must be resolved within a reasonable time. However, the traditional method of cutting TFT-LCD plates cannot resolve the CSP effectively.

Cutting various TFT-LCD plates from a glass substrate of limited dimensions (i.e., stock) is a well-known assortment problem (Beasley, 1985; Chen, Sarin, & Balasubramanian, 1993; Li & Chang, 1998; Li, Chang, & Tsai, 2002; Li, Tsai, & Hu, 2003; Lin, 2006), whereas minimizing the amount of stock utilized and satisfying customer demand is a classic CSP (Chambers & Dyson, 1976; Correia, Oliveira, & Ferreira, 2004; Demir, 2008; Holthaus, 2002; Tsai, Hsieh, & Huang, 2009; Wagner, 1999).

CSPs have been studied in various applications (Cui & Lu, 2009; De Queiroz, Miyazawa, Wakabayashi, & Xavier, 2012; Zheng, Ren, Ge, Qiu, & Liu, 2011) such as crosscutting rectangular products from wood stocks (Reinders, 1992), cutting TFT-LCD plates from glass substrates (Tsai et al., 2009), placing all devices in a system-on-a-chip circuit (Li, Ma, Xu, Want, & Hong, 2009; Tang & Yao, 2007), and overseas container shipping services (Chen, Lee, & Shen, 1995; Pisinger, 2002; Wang, Li, & Levy, 2008). The problem has generated a great deal of interest because minimizing the amount of stock would significantly reduce production overheads and increase a company's competitiveness.

Hinxman (1980) first classified cutting stock problems as one-dimensional, 11/2-dimensional, and two-dimensional problems. The approaches for solving CSPs can be classified as heuristic and deterministic-based methods. Beaslay (1985) proposed a heuristic-based method that uses an integer model and a heuristic algorithm to solve a two-dimensional CSP in the guillotine industry, whereas Jakobs (1996) designed a genetic algorithm (GA) to improve the efficiency of deterministic methods. Leung, Chan, and Troutt (2003) presented a mixed simulated annealing-genetic algorithm to accelerate the solution time of CSPs. Umetani, Yagiura, and Ibaraki (2003) utilized meta-heuristics and adaptive pattern generation techniques to minimize the number of one-dimensional CSPs with different patterns. Subsequently, Gradisar and Trkman

* Corresponding author. Tel.: +886 35712121/57416; fax: +886 3 5723792.
*E-mail addresses:* bach0809@gmail.com (H.-C. Lu), yaohuei.huang@gmail.com (Y.-H. Huang), andy@mail.lhu.edu.tw (K.-A. Tseng).

0360-8352/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cie.2013.02.009

(2005) proposed a combined approach to solve general one-dimensional CSPs, whereas Lin (2006) introduced a genetic algorithm that incorporates a novel encoding schema in a random packing process to solve two dimensional CSPs (Gonçalves, 2007; Jakobs, 1996; Kroger, 1995). Although all heuristic-based methods guarantee a solution within a reasonable time, such solution may not be optimal.

As regards the deterministic-based methods, Page (1975) proposed a two-dimensional dynamic programming model for cutting rectangular steel plates. Chen et al. (1993) constructed a mixed integer programming (MIP) model to solve an assortment problem that involved placing a set of small different-sized rectangles, in a non-overlapping formulation, within a large rectangular minimum area. To improve Chen et al.'s nonlinear program with non-overlapping constraints, Li and Chang (1998) proposed a bilinear objective model that linearizes the objective function (i.e., $xy$) and reduces the number of binary variables for the non-overlapping constraints. Li et al. (2002) utilized a logarithm-based decomposition technique to solve the bilinear objective function. Subsequently, Li et al. (2003) converted the original assortment problem into a number of sub-problems by dividing the objective value into several intervals, after which they solved the related sub-problems with a set of personal computers using parallel network technology. This approach demonstrates that when the solution time is unrestricted, deterministic-based methods can exploit the analytical properties of the problems to generate a sequence of points that converge to an optimal solution (Chen et al., 1993; Li & Chang, 1998; Li et al., 2002; Li et al., 2003).

Deterministic-based methods explored well-known algorithms such as the simplex method for linear programs (Dantzig, 1963), and the branch-and-bound method for mixed integer programs (Land & Doing, 1960). However, in the branch-and-bound method, the CSP with all its extensions and variants has been classified as NP-hard (Garey & Johnson, 1979; Lai & Chan, 1997), thus making the derivation of an optimum solution within a reasonable time impossible. To address this problem, this study utilizes a GA to identify the possible cutting patterns and proposes a high-speed corner arrangement (CA) method to verify such patterns. We then use a mixed integer program to construct a production plan (PP) model and minimize the number of glass substrates.

In this study, we regard each glass substrate as an item of cutting stock with fixed dimensions. The substrate is then cut into TFT-LCD plates according to customer requirements. Considering that the CSP in the TFT-LCD industry tries to minimize the number of glass substrates, we utilize an integrated algorithm that (a) implements a GA to refine the fitness value of the cutting patterns, (b) verifies all possible cutting patterns with a corner arrangement method, and (c) utilizes a mixed integer program to minimize the number of glass substrates needed to satisfy customer requirements. When we compared the proposed method with current method (Tsai et al., 2009), we observed that:

(i) Tsai et al.'s optimization algorithm only performed well on small-scale problems. That is, Model 2 in their algorithm is unsuitable for verifying possible cutting patterns in real world cases, especially when the number of small rectangles becomes large.
(ii) Garey and Johnson (1979) also proved that the cutting stock problem is NP-hard solved by MIP. That is, real world cases in the TFT-LCD industry are all large-scale CSPs, which cannot be solved optimally by deterministic methods because the computation time grows exponentially.

The proposed algorithm seeks and verifies possible patterns in a short time to reduce waste of the stocks, and a mixed integer program constructs a PP model to minimize the number of glass substrates needed to satisfy customer demand. The key advantages of the proposed integrated algorithm are as follows:

(i) All possible cutting patterns are quickly verified by implementing the proposed CA method.
(ii) The quality of the cutting patterns is guaranteed by the fitness function, and the algorithm can find a solution within a reasonable amount of time.

The remainder of this paper is organized as follows. Section 2 introduces three methods used to solve the CSP: a GA for coding strings of placement in orders also called possible cutting patterns; a CA method that verifies possible cutting patterns; and a PP model that minimizes the amount of stock required to fulfill orders. Section 3 describes the proposed integrated algorithm for solving the CSP in the TFT-LCD industry. Numerical examples are given in Section 4 to demonstrate the efficacy of the proposed algorithm. Section 5 contains some concluding remarks.

## 2. Three methods

Three methods are constructed to solve the CSP in the TFT-LCD industry. These methods identify and verify possible cutting patterns and minimize the amount of stocks required to fulfill orders. First, we define the notations used in the remainder of this paper.

Notations:

| | |
|---|---|
| (Width, Length) | The width and length of the glass substrate |
| $(p_i, q_i)$ | The length and width of the $i$th TFT-LCD plate |
| $i$ | The $i$th TFT-LCD plate |
| $I$ | The number of TFT-LCD plates |
| $f_g^i$ | The $i$th TFT-LCD plate in the $g$th cutting pattern |
| $F_g$ | The $g$th feasible cutting pattern that identifies different TFT-LCD plates |
| $N$ | The initial length of the chromosome |
| $n_g$ | The number of small rectangles cut from a glass substrate based on the $g$th pattern |
| $\pi_g$ | The $g$th cutting pattern as a permutation (i.e., chromosome) |
| $\pi_g^j$ | The $j$th small rectangles in the $g$th permutation (i.e., chromosome) |
| $Q_g$ | The $g$th set of feasible cutting patterns that mark the coordinates of the small rectangles |
| $C_g$ | The $g$th set of corners |
| $j$ | The indicator of the small rectangle $j$ |
| $k$ | The indicator of the corner $k$ |
| $M$ | The number of cutting patterns |
| $a_g^j$ | The cutting element $a_g^j = [x_g^j, y_g^j, p_g^j, q_g^j]$, where $(x_g^j, y_g^j)$ is the coordinate of the small rectangle $j$ placed along the $x$-axis and $y$-axis from the original point $(p_g^j, q_g^j)$ |
| $c_g^k$ | The corner element $c_g^k = [x_g^k, y_g^k, w_g^k, l_g^k]$, where $(x_g^k, y_g^k)$ is the coordinate of the $k$th corner space, and $(w_g^k, l_g^k)$ denotes the $k$th dimension of the corner space of the $g$th set |
| $(p_g^j, q_g^j)$ | The component of $\pi_g^j$, which expresses the width and the length of the small rectangles $j$ in the $g$th pattern |

### 2.1. Method 1 – GA

A GA is a heuristic method that mimics natural selection (Gonçalves, 2007; Kroger, 1995; Lin, 2006; Wagner, 1999) through the processes of reproduction, crossover, and mutation. Initially, all possible cutting patterns are unique and randomly generated based on the sizes of the produced TFT-LCD plates. The patterns are then refined by GA iteratively.

The length of the chromosome $N$ is determined by the initial step in the GA and considers as many kinds of TFT-LCD plates as possible, such that

$$N = \mathrm{MAX}\left\{\left\lceil\frac{Length \times Width}{p_i \times q_i}\right\rceil, \quad \forall\, i = 1,\ldots,I\right\}.$$

In this study, the GA method uses the roulette wheel selection process to select two chromosomes from the population for reproduction based on the proportion of cutting patterns. The steps of the process are as follows:

(i) *Decoding and permutation*: The component in the permutation, $\pi_g^j$, is randomly generated from a uniform distribution based on $\pi_g^j \in \{(p_1, q_1), (p_2, q_2), \ldots, (p_I, q_I)\}$ for $j = 1, \ldots, N$, and the permutations $\pi_g = (\pi_g^1, \pi_g^2, \ldots, \pi_g^N)$ are also denoted as a chromosome for $g = 1, \ldots, M$. Each $\pi_g$ in a chromosome is unique. The advantage of the genetic permutation is that it facilitates the random generation of a new permutation as a sorting sequence, which is refined in the crossover and mutation steps.

(ii) *Crossover*: The crossover operator generates a new offspring. Specifically, the roulette wheel selection process combines two chromosomes to generate a new one. Let $N = 6$. We choose a pair of chromosomes and select two cutting points for crossover, as shown in Fig. 1.

The elements between the two cutting points of the first parent are included in the crossover operation, and the elements between the two cutting points of the second parent are excluded. We then incorporate these elements into the new child produced by parent 1 (i.e., $g = 1$) and parent 2 (i.e., $g = 2$). The concept is illustrated in Fig. 2.

(i) *Mutation*: The mutation operator exchanges two components in the new child if a random probability value is greater than a threshold value, as shown in Fig. 3.

The operator then exchanges $q_j$ and $p_j$ when another random probability value is greater than a threshold value, which indicates that a rectangle is rotated 90° (Fig. 4).

In the initial setting of the GA, we randomly and uniquely generate $M$ possible cutting patterns. We then utilize the GA to refine the derived cutting patterns, after which each possible cutting pattern is verified by applying method 2 – CA.

### 2.2. Method 2 – CA

To verify possible cutting patterns within a reasonable time, we propose a solution called the CA method. The method identifies a rectangular domain space and marks the bottom left-hand point of the space as a corner. After the space is marked as a cutting small rectangle (i.e., TFT-LCD plate), the remaining space is divided into two sub-spaces, wherein the other smaller rectangles can be

**CA algorithm**

**Initial:**

(i): Possible cutting patterns are expressed as $\pi_g = (\pi_g^1, \pi_g^2, \ldots, \pi_g^N)$ (e.g., $\pi_g^1 = (p_g^1, q_g^1)$).

(ii): Let $n_g = 0, j = 1, k = 1, x_g^k = 0, y_g^k = 0, w_g^k = Width, l_g^k = Length, C_g = \{c_g^1\}$, and $c_g^1 = (x_g^1, y_g^1, w_g^1, l_g^1)$.

(iii): Let $Q_g = \varnothing$, where "$\varnothing$" denotes that no small rectangle has been cut.

**Arrangement:**

(i): If the small rectangle $j$ can be fully covered (see Definition 1) by the space of the corner $c_g^{k'}$ (where $p_g^j$ is parallel to the $x$-axis of the corner $k', p_g^j \leqslant w_g^{k'}$, and $k' \in \{1, \ldots, k\}$), the following operations are executed:

(a) $n_g = n_g + 1$.

(b) Mark the small rectangle $j$ in the corner $c_g^{k'}$. That is, $a_g^{n_g} = (x_g^{k'}, y_g^{k'}, p_g^j, q_g^j)$, and add $a_g^{n_g}$ to $Q_g$.

(c) Update the corner $c_g^{k'} = (x_g^{k'} + p_g^j, y_g^{k'}, w_g^{k'} - p_g^j, q_g^j)$.

(d) Insert a new corner $c_g^{k+1} = (x_g^{k'}, y_g^{k'} + q_g^j, w_g^{k'}, l_g^{k'} - q_g^j)$ into $C_g$, and go to step (iii).

(ii): If the small rectangle $j$ can be fully covered (see Definition 1) in the corner $c_g^{k'}$ (where $p_g^j$ is parallel to the y-axis of the corner $k', p_g^j \leqslant l_g^k$, and $k' \in \{1, \ldots, k\}$), then the following operations are executed:

(a) $n_g = n_g + 1$.

(b) Mark the small rectangle $a_g^{n_g} = (x_g^{k'}, y_g^{k'}, q_g^j, p_g^j)$ in the corner $c_g^{k'}$, and add $a_g^{n_g}$ to $Q_g$.

(c) Update the corner $c_g^{k'}$ with $c_g^{k'} = (x_g^{k'} + q_g^j, y_g^{k'}, w_g^{k'} - q_g^j, p_g^j)$.

(d) Insert a new corner $c_g^{k+1} = (x_g^{k'}, y_g^{k'} + p_g^j, w_g^{k'}, l_g^{k'} - p_g^j)$ into $C_g$, and go to step (iii).

(iii): Let $j = j + 1$ and go to step (i) if $j \leqslant N$.

**End**

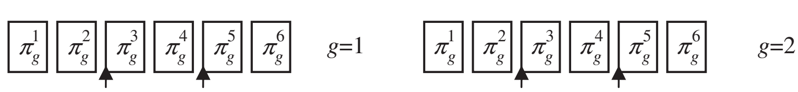


**Fig. 1.** Random selection of two cutting points in the crossover operator.

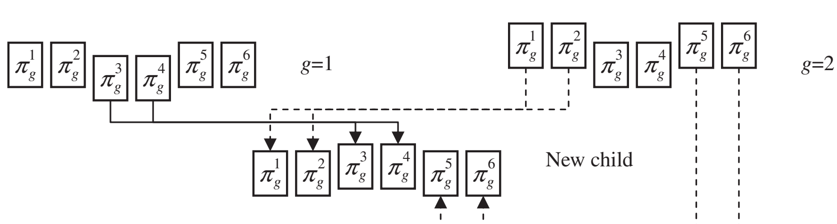


**Fig. 2.** Crossover operation generates a new child.

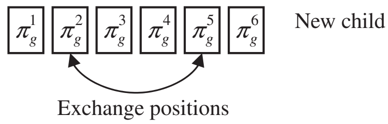


**Fig. 3.** The operator exchanges two components in the new child.

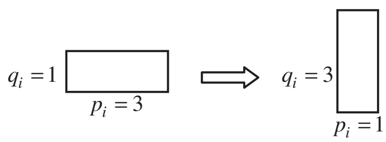


**Fig. 4.** A rectangle is rotated 90°.

placed. The objective is to use the least square measure of the glass substrate to generate the needed small rectangles. The steps of the CA algorithm are as follows:

A simplified demonstration of the CA algorithm is shown in Fig. 5.

**Definition 1.** A feasible corner space can be used to fully cut a required small rectangle (i.e., TFT-LCD plate) that satisfies the following conditions:

(i) The $j$th small rectangle, $(p_g^j, q_g^j)$ can be covered by a feasible corner $k'$, where $k' \in \{1, 2, \ldots, k\}$ satisfies either
- $l_g^{k'} \geqslant p_g^j$ and $w_g^{k'} \geqslant q_g^j$; or
- $l_g^{k'} \geqslant q_g^j$ and $w_g^{k'} \geqslant p_g^j$.

(ii) If condition (i) is satisfied, we obtain a set of feasible corner spaces $C_g$. Considering the minimal trim-loss issue (i.e., in space with minimum waste), a corner space $k$' is the best choice, where $k' = \mathrm{Max}\{\frac{p_g^j + q_g^j}{l_g^k + w_g^k} \leqslant 1,\ \forall C_g\}$. Otherwise, no feasible corner spaces exist.

Given that the fitness values of each feasible cutting pattern must be verified (i.e., utility rate), we have Remark 1 below.

**Remark 1.** The fitness value of the feasible cutting pattern is calculated by following function:

$$Fitness(g) = \frac{\sum_{i=1}^{n_g} p_g^i \times q_g^i}{Length \times Width} \times 100\%.$$

### 2.3. Method 3 – PP model

After the feasible cutting patterns are verified by the CA algorithm and refined by the GA iteratively, these patterns are prepared for the optimal production scheme. The operations are discussed as follows:

**Initial production plan:**
(i) $f_g^i = 0$ for $g = 1, \ldots, M$ and $i = 1, \ldots, I$.
(ii) If the cutting element $a_g^j$ in the set $Q_g$ belongs to the $i$th TFT-LCD plate, then $f_g^i = f_g^i + 1$ for $g = 1, \ldots, M$, $j = 1, \ldots, n_g$ and $i = 1, \ldots, I$.

Let $F_g = (f_g^1, f_g^2, \ldots, f_g^I)$ be the gth cutting pattern. In addition, let $D_i$ denote the order quantity of the $i$th TFT-LCD plate, and let the decision variable $U_g$ indicate the number of TFT-LCD plates that can be cut from the gth feasible cutting pattern. To minimize the number of glass substrates required to fulfill customer orders, the optimal PP model is formulated as follows:

**PP Model**

$$\mathrm{Min} \sum_{g=1}^{M} U_g$$

,

$$\text{s.t.} \sum_{g=1}^{M} f_g^i U_g \geqslant D_i \quad \text{for } i = 1, \ldots, I,$$

where $U_g \in \mathbb{Z}^+$.

## 3. Integrated algorithm

The proposed integrated algorithm explores three methods. First, the GA method refines $M$ possible cutting patterns. Second, the CA method verifies each possible cutting pattern. Finally, the PP method guarantees that the number of glass substrates is minimal.

The detail steps of the algorithm are as follows:

**Input values:** {*iterations*, *threshold values*, (*Length*, *Width*), $M$, $(p_i, q_i)$ and $D_i$ for $i = 1, \ldots, I$}
**Preprocessing:**
{
**Step 1:** $current\_ite$=0, $j = 1$, $k = 1$, $x_g^1 = 0$, $y_g^1 = 0$, $w_g^1 = Width$, $l_g^1 = Length$, $C_g = \{c_g^1\}$, $c_g^1 = (x_g^1, y_g^1, w_g^1, l_g^1)$, $Q_g = \emptyset$, $N = \mathrm{MAX}\left\{\left\lceil \frac{Length \times Width}{p_i \times q_i} \right\rceil \text{ for } i = 1, \ldots, I\right\}$, and $\pi_g = (\pi_g^1, \pi_g^2, \ldots, \pi_g^N)$ for $g = 1, \ldots, M$.
**Step 2:** Based on a uniform distribution, randomly generate $\pi_g^j$, where $\pi_g^j \in \{(p_1, q_1), (p_2, q_2), \ldots, (p_I, q_I)\}$ for $j = 1, \ldots, N$, to derive a unique permutation $\pi_g = (\pi_g^1, \pi_g^2, \ldots, \pi_g^N)$ for $g = 1, \ldots, M$ (i.e., Chromosomes).
}
**Process:**
{
**Step 3:** Execute the CA method to verify each fitness value of the $M$ feasible cutting patterns.
**Step 4:** Execute the main process of the GA method to generate $M_2$ new children (i.e., $M + M_2$ possible cutting patterns), such that *current_ite* = *current_ite* + 1.
**Step 5:** Execute the CA method to verify the fitness values of the $M_2$ new children.
**Step 6:** Delete the worst feasible cutting patterns of the last $M_2$ children according to their fitness values.
**Step 7:** Go to Step 4 until *current_ite* > *iterations* or each fitness value of the feasible cutting pattern > *threshold value.*
**Step 8:** Execute the PP method, including the preprocessing step and the PP model.
}
**Output:** {The optimal production combination $(U_1, U_2, \ldots, U_M)$}

The flowchart of the proposed algorithm for generating and verifying the cutting patterns and outputting of the optimal production plan is shown in Fig. 6.

## 4. Numerical examples and experiments

We applied the integrated algorithm to real CSPs in Taiwan's TFT-LCD industry. The algorithm was implemented in Java programming language (version 6), which also embeds the MIP solution engine of ILOG CPLEX 11 (ILOG 2008) to solve the PP model and runs on a PC equipped with an Intel Pentium® Dual-Core

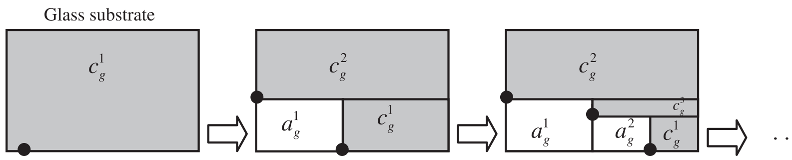


**Fig. 5.** Demonstration of the CA algorithm.

2.8 GHz CPU and 2 GB RAM. Identifying and verifying possible cutting patterns are recognized to be the most difficult aspects of solving CSPs. Our experiments show that the proposed algorithm is more efficient than the current method.

Additionally, we compare the performance of the proposed algorithm with that of the following two approaches for placement procedures (i.e., verifying a cutting pattern).

(i) *Batch production method* – The current production approach used in the TFT-LCD industry is batch production, whereby each glass substrate is cut into TFT-LCD plates of one size only. Generally, the method is based on rule of thumb.
(ii) *Integer programming method* – Tsai et al. (2009) proposed an MIP model (Model 2 in their approach) to find feasible cutting patterns. Although the solution of the proposed algorithm is a global optimum in seeking each pattern, it is inefficient for large-scale problems.

We then compare the performance of our approach with that of Tsai et al.'s MIP method in terms of verifying each possible cutting

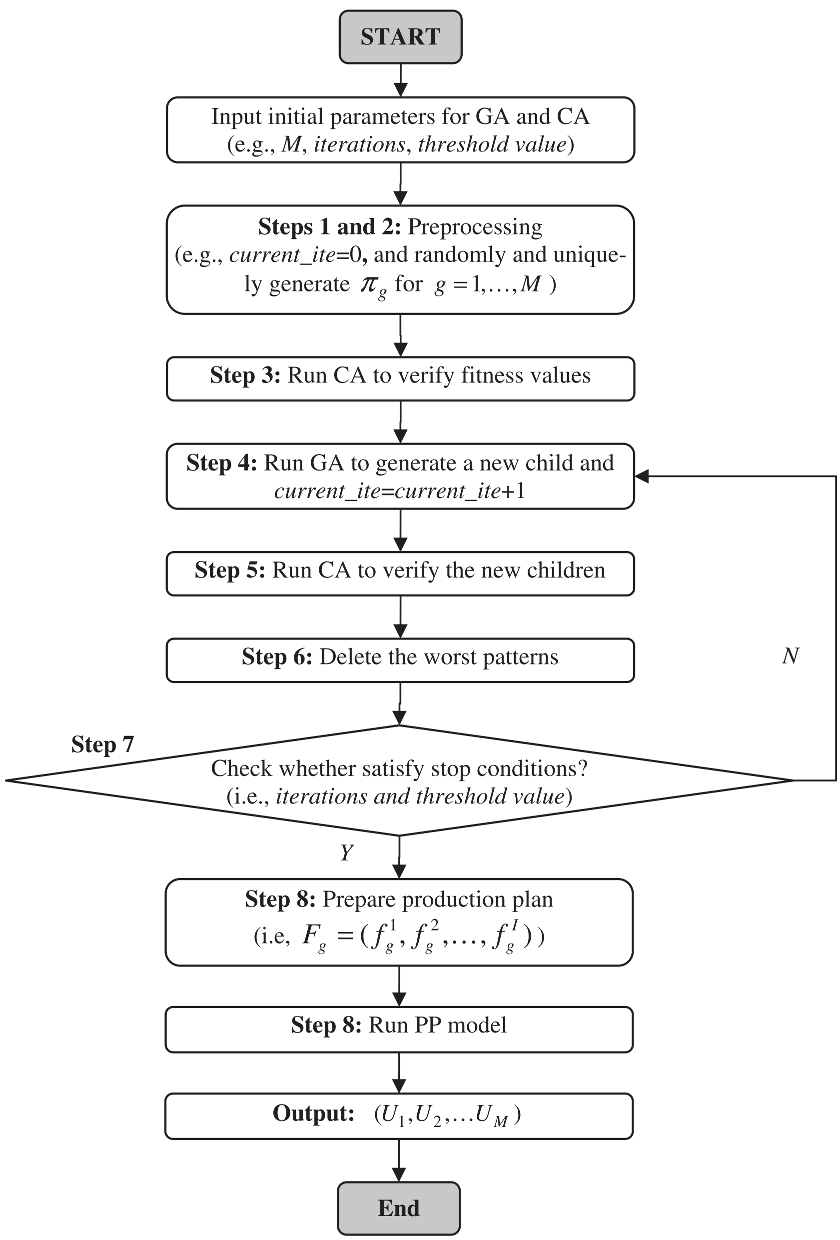


**Fig. 6.** Flowchart of the proposed algorithm.

**Table 1**
Data for different-sized problems.

| $n = 6$ | | $n = 8$ | | $n = 20$ | |
|---|---|---|---|---|---|
| Type | Quantity | Type | Quantity | Type | Quantity |
| (100,62) cm | 6 | (90,56) cm | 8 | (17,10) cm | 7 |
| | | | | (25,12) cm | 6 |
| | | | | (85,54) cm | 2 |
| | | | | (100,62) cm | 5 |
| $n = 90$ | | $n = 120$ | | $n = 235$ | |
| (10,6) cm | 29 | (10,6) cm | 64 | (10,6) cm | 228 |
| (17,10) cm | 23 | (17,10) cm | 49 | (90,56) cm | 4 |
| (25,12) cm | 30 | (90,56) cm | 3 | (100,62) cm | 3 |
| (85,54) cm | 8 | (100,62) cm | 4 | | |

**Table 2**
Comparison of the proposed method and the integer programming method.

| # Of products ($n$) | Proposed method | | | Integer programming method | |
|---|---|---|---|---|---|
| | Best fitness (%) | Average fitness (%) | Average CPU time (s) | Best fitness | CPU time (s) |
| 6 | 99 | 98 | 5 | 99% | 1850 |
| 8 | 99 | 98 | 6 | 99% | 7500 |
| 20 | 97 | 96 | 7 | Infeasible | >36,000 |
| 90 | 98 | 95 | 9 | Infeasible | >36,000 |
| 120 | 98 | 93 | 10 | Infeasible | >36,000 |
| 235 | 97 | 93 | 10 | Infeasible | >36,000 |

pattern. Thereafter, we discuss a real-world CSP in Taiwan's TFT-LCD industry to minimize the amout of stocks utilized.

### 4.1. Comparison with the integer programming method

Efficient verification of each possible cutting pattern is an important issue in this study. Therefore, the solution quality might be sacrificed because the computation time is unacceptable when the problem is large. Table 1 shows different-sized problems observed in the 10th generation (10G) TFT-LCD industry, as well as the results of the verification of six cutting patterns with the proposed CA method and Model 2 of Tsai et al.'s method. The latter is solved by CPLEX (2009) software. The proposed algorithm ran each problem for 500 iterations in 30 rounds (with $n = 6$, 8, 20, 90, 120 and 235).

The results demonstrate that the proposed algorithm is considerably more efficient than Tsai et al.'s integer programming method, which derives optimal solutions for problems when $n = 6$ and 8, but fails to obtain feasible solutions for problems when $n \geqslant 20$. Meanwhile, the CPU time required to reach optimality increases rapidly with the problem size, and the method cannot determine the optimal solution within 36,000 s (i.e., the solution time >10 h) when the problem size exceeds 20. The results in Table 2 show that the proposed method is more efficient in terms of computation time because the CA method is efficient in evaluating cutting patterns.

### 4.2. Experiments on CSPs in Taiwan's 10G TFT-LCD industry

We conducted experiments on CSPs of different-sized products in Taiwan's 10G TFT-LCD industry. Information about the TFT-LCDs is available from the following Web sites:

**Table 3**
Dimensions of the glass substrates for the 20 orders.

| Items | | Order # | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Product # | $(W,L)_{\text{cm}}$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | (15,20) | 350 | 1150 | 1000 | 1500 | 2450 | 4000 | 5000 | 13,000 | 30,000 | 95,000 |
| 2 | (90,56) | 215 | 1500 | 1000 | 2500 | 1670 | 3000 | 2000 | 17,500 | 30,000 | 77,000 |
| 3 | (93,60) | 205 | 1250 | 1000 | 3000 | 1300 | 2000 | 3000 | 15000 | 50,000 | 85,000 |
| 4 | (99,63) | 600 | 1600 | 2000 | 2500 | 1120 | 2000 | 4500 | 45,000 | 50,000 | 65,000 |
| 5 | (95,66) | 500 | 2500 | 1000 | 2000 | 1850 | 1000 | 1000 | 28,000 | 50,000 | 90,000 |
| 6 | (98,64) | 810 | 2800 | 2000 | 2000 | 1600 | 4000 | 2500 | 20,000 | 40,000 | 85,000 |
| 7 | (100,62) | 990 | 3900 | 1000 | 2000 | 1350 | 3000 | 3300 | 33,500 | 40,000 | 80,000 |
| 8 | (108,72) | 555 | 2100 | 2000 | 2000 | 1990 | 3000 | 1700 | 35,000 | 40,000 | 70,000 |
| 9 | (126,82) | 990 | 2900 | 1000 | 2000 | 1330 | 3000 | 6000 | 12,500 | 40,000 | 90,000 |
| | | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 1 | (15,20) | 30,000 | 50,000 | 70,000 | 90,000 | 100,000 | 300,000 | 400,000 | 500,000 | 500,000 | 800,000 |
| 2 | (90,56) | 30,000 | 50,000 | 70,000 | 90,000 | 100,000 | 300,000 | 400,000 | 500,000 | 500,000 | 800,000 |
| 3 | (93,60) | 60,000 | 50,000 | 70,000 | 90,000 | 100,000 | 300,000 | 400,000 | 500,000 | 500,000 | 800,000 |
| 4 | (99,63) | 60,000 | 50,000 | 70,000 | 90,000 | 100,000 | 300,000 | 400,000 | 600,000 | 700,000 | 800,000 |
| 5 | (95,66) | 70,000 | 60,000 | 70,000 | 90,000 | 100,000 | 500,000 | 400,000 | 600,000 | 700,000 | 800,000 |
| 6 | (98,64) | 70,000 | 60,000 | 80,000 | 90,000 | 200,000 | 500,000 | 400,000 | 600,000 | 700,000 | 850,000 |
| 7 | (100,62) | 70,000 | 60,000 | 80,000 | 90,000 | 200,000 | 500,000 | 400,000 | 700,000 | 800,000 | 850,000 |
| 8 | (108,72) | 90,000 | 90,000 | 80,000 | 90,000 | 200,000 | 500,000 | 400,000 | 700,000 | 800,000 | 850,000 |
| 9 | (126,82) | 90,000 | 90,000 | 80,000 | 90,000 | 200,000 | 500,000 | 400,000 | 700,000 | 800,000 | 850,000 |

**Table 4**
Feasible cutting patterns derived by the proposed algorithm.

| $g$ | $f_g^1$ | $f_g^2$ | $f_g^3$ | $f_g^4$ | $f_g^5$ | $f_g^6$ | $f_g^7$ | $f_g^8$ | $f_g^9$ | # Of products | Fitness (%) | $g$ | $f_g^1$ | $f_g^2$ | $f_g^3$ | $f_g^4$ | $f_g^5$ | $f_g^6$ | $f_g^7$ | $f_g^8$ | $f_g^9$ | # Of products | Fitness (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 6 | 71 | 26 | 38 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 46 | 96 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 8 | 72 | 27 | 37 | 1 | 1 | 1 | 2 | 0 | 4 | 2 | 0 | 48 | 93 |
| 3 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 8 | 0 | 11 | 93 | 28 | 36 | 0 | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 49 | 96 |
| 4 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 0 | 0 | 11 | 79 | 29 | 37 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 49 | 98 |
| 5 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 11 | 79 | 30 | 39 | 3 | 3 | 4 | 1 | 1 | 0 | 0 | 0 | 51 | 93 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 11 | 78 | 31 | 41 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 52 | 94 |
| 7 | 0 | 3 | 3 | 2 | 1 | 0 | 0 | 0 | 3 | 12 | 94 | 32 | 43 | 0 | 0 | 11 | 0 | 0 | 0 | 0 | 0 | 54 | 94 |

(continued on next page)

**Table 4** (*continued*)

| $g$ | $f_g^1$ | $f_g^2$ | $f_g^3$ | $f_g^4$ | $f_g^5$ | $f_g^6$ | $f_g^7$ | $f_g^8$ | $f_g^9$ | # Of products | Fitness (%) | $g$ | $f_g^1$ | $f_g^2$ | $f_g^3$ | $f_g^4$ | $f_g^5$ | $f_g^6$ | $f_g^7$ | $f_g^8$ | $f_g^9$ | # Of products | Fitness (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 0 | 0 | 0 | 8 | 3 | 0 | 0 | 0 | 1 | 12 | 91 | 33 | 43 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 54 | 93 |
| 9 | 0 | 5 | 0 | 0 | 0 | 0 | 4 | 0 | 3 | 12 | 93 | 34 | 46 | 2 | 4 | 0 | 5 | 1 | 0 | 0 | 0 | 58 | 96 |
| 10 | 0 | 0 | 5 | 2 | 0 | 5 | 0 | 0 | 0 | 12 | 83 | 35 | 48 | 1 | 6 | 2 | 2 | 1 | 0 | 0 | 0 | 60 | 97 |
| 11 | 0 | 0 | 0 | 0 | 12 | 0 | 0 | 0 | 0 | 12 | 87 | 36 | 48 | 5 | 0 | 0 | 0 | 7 | 0 | 0 | 0 | 60 | 96 |
| 12 | 0 | 0 | 7 | 1 | 0 | 5 | 0 | 0 | 0 | 13 | 88 | 37 | 48 | 11 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 61 | 95 |
| 13 | 0 | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 15 | 87 | 38 | 54 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 5 | 62 | 97 |
| 14 | 0 | 0 | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 15 | 96 | 39 | 52 | 3 | 6 | 1 | 2 | 0 | 0 | 0 | 0 | 64 | 95 |
| 15 | 13 | 1 | 1 | 1 | 0 | 2 | 1 | 0 | 4 | 23 | 93 | 40 | 51 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 65 | 99 |
| 16 | 18 | 0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 32 | 96 | 41 | 54 | 0 | 0 | 0 | 9 | 2 | 0 | 0 | 0 | 65 | 98 |
| 17 | 23 | 3 | 2 | 4 | 4 | 0 | 0 | 0 | 0 | 36 | 96 | 42 | 58 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 4 | 66 | 96 |
| 18 | 27 | 1 | 3 | 0 | 1 | 2 | 1 | 1 | 2 | 38 | 96 | 43 | 66 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 72 | 94 |
| 19 | 24 | 8 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 38 | 97 | 44 | 62 | 1 | 2 | 8 | 0 | 0 | 0 | 0 | 0 | 73 | 97 |
| 20 | 28 | 2 | 2 | 0 | 2 | 1 | 1 | 1 | 2 | 39 | 96 | 45 | 71 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 79 | 96 |
| 21 | 29 | 0 | 0 | 0 | 12 | 0 | 0 | 0 | 0 | 41 | 97 | 46 | 72 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 79 | 96 |
| 22 | 30 | 1 | 2 | 1 | 0 | 4 | 4 | 0 | 0 | 42 | 94 | 47 | 74 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 84 | 97 |
| 23 | 32 | 0 | 4 | 2 | 1 | 2 | 0 | 1 | 1 | 43 | 94 | 48 | 77 | 9 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 89 | 98 |
| 24 | 33 | 1 | 0 | 1 | 3 | 1 | 6 | 0 | 0 | 45 | 96 | 49 | 87 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 99 | 100 |
| 25 | 36 | 4 | 1 | 0 | 0 | 1 | 0 | 1 | 3 | 46 | 94 | 50 | 285 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 285 | 98 |

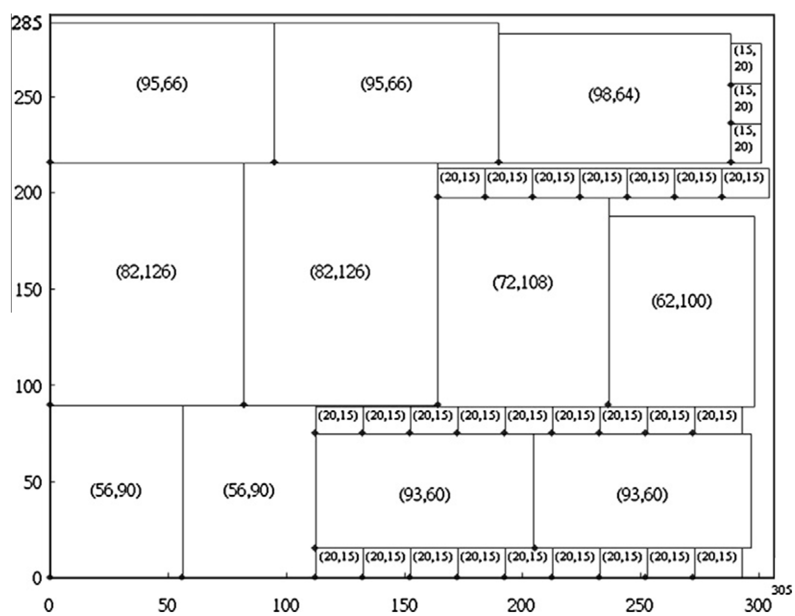


**Fig. 7.** Solution for pattern #20 derived by the proposed algorithm.

**Table 5**
Coordinates of pattern #20 derived by the proposed algorithm.

| # | $x$-Axis | $y$-Axis | $p_i$ | $q_i$ | # | $x$-Axis | $y$-Axis | $p_i$ | $q_i$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 288 | 216 | 15 | 20 | 21 | 132 | 75 | 20 | 15 |
| 2 | 288 | 236 | 15 | 20 | 22 | 152 | 75 | 20 | 15 |
| 3 | 288 | 256 | 15 | 20 | 23 | 172 | 75 | 20 | 15 |
| 4 | 112 | 0 | 20 | 15 | 24 | 192 | 75 | 20 | 15 |
| 5 | 132 | 0 | 20 | 15 | 25 | 212 | 75 | 20 | 15 |
| 6 | 152 | 0 | 20 | 15 | 26 | 232 | 75 | 20 | 15 |
| 7 | 172 | 0 | 20 | 15 | 27 | 252 | 75 | 20 | 15 |
| 8 | 192 | 0 | 20 | 15 | 28 | 272 | 75 | 20 | 15 |
| 9 | 212 | 0 | 20 | 15 | 29 | 0 | 0 | 56 | 90 |
| 10 | 232 | 0 | 20 | 15 | 30 | 56 | 0 | 56 | 90 |
| 11 | 252 | 0 | 20 | 15 | 31 | 236 | 90 | 62 | 100 |
| 12 | 272 | 0 | 20 | 15 | 32 | 164 | 90 | 72 | 108 |
| 13 | 164 | 198 | 20 | 15 | 33 | 0 | 90 | 82 | 126 |
| 14 | 184 | 198 | 20 | 15 | 34 | 82 | 90 | 82 | 126 |
| 15 | 204 | 198 | 20 | 15 | 35 | 112 | 15 | 93 | 60 |
| 16 | 224 | 198 | 20 | 15 | 36 | 205 | 15 | 93 | 60 |
| 17 | 244 | 198 | 20 | 15 | 37 | 0 | 216 | 95 | 66 |
| 18 | 264 | 198 | 20 | 15 | 38 | 95 | 216 | 95 | 66 |
| 19 | 284 | 198 | 20 | 15 | 39 | 190 | 216 | 98 | 64 |
| 20 | 112 | 75 | 20 | 15 | | | | | |

- STPI: http://cdnet.stpi.org.tw/techroom/market/eedisplay/eedisplay290.htm.
- Mobile: http://www.mobile01.com/topicdetail.php?f=350&t=874736&p=1.

In the experiments, the dimensions of the glass substrate are 305 cm × 285 cm in the 10G facility, and the PP involves the production of nine kinds of TFT-LCD plates to satisfy 20 orders. The details of the orders are listed in Table 3.

The current state-of-the-art theory on GAs does not provide information about parameter setting (see Gonçalves, 2007). In our experience, GAs based on the same evolutionary strategy have the following initial parameters.

- The number of possible cutting patterns is 50 (i.e., $M = 50$).
- Rotation probability: 0.7 to 0.8 (i.e., exchange $p_j$ with $q_j$).
- Stopping criteria: Iterations > 2000 or *Fitness*($g$) > 90% for all $g$.
- Runs 30 rounds.

The solutions solved by the proposed algorithm are listed in Table 4. Take pattern #1 in the Table 4 as an example. The numbers

**Table 6**
Number of cutting patterns used by the proposed method in the10G TFT-LCD experiments.

| $u_g$ | Order # | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 1 | 0 | 0 | 1 | 7 | 4 | 3 | 78 | 1 | 3 | 0 | 2 | 0 | 4 | 1 | 0 | 0 | 0 | 0 | 5 | 0 |
| 2 | 231 | 620 | 81 | 84 | 82 | 355 | 1249 | 222 | 6011 | 12,574 | 21,936 | 16,914 | 11,002 | 9531 | 32,713 | 96,874 | 42,358 | 120,997 | 153,128 | 102,675 |
| 3 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 1 | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1776 | 0 | 0 | 20544 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 9 | 1 | 191 | 31 | 0 | 0 | 103 | 0 | 3740 | 2847 | 3056 | 4310 | 0 | 4185 | 3090 | 0 | 15,622 | 13,755 | 53,039 | 71,033 | 36,436 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 21 | 0 | 0 | 39 | 111 | 145 | 45 | 0 | 526 | 2638 | 6013 | 0 | 4852 | 4010 | 5363 | 6620 | 37,761 | 23,836 | 31,632 | 35,080 | 46,866 |
| 22 | 2 | 3 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 23 | 0 | 122 | 0 | 166 | 130 | 284 | 0 | 1438 | 2284 | 5156 | 1983 | 0 | 4890 | 0 | 0 | 26,421 | 27,412 | 27,725 | 29,337 | 55,291 |
| 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 14,489 | 0 | 0 | 0 | 0 |
| 26 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 30 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 31 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 32 | 0 | 0 | 0 | 0 | 1 | 0 | 31 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 34 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 35 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 36 | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 37 | 20 | 0 | 142 | 206 | 35 | 50 | 307 | 3489 | 3267 | 2888 | 0 | 0 | 3100 | 5451 | 0 | 1 | 24,236 | 20,433 | 21,979 | 42,767 |
| 38 | 69 | 262 | 250 | 250 | 249 | 375 | 213 | 4375 | 5000 | 8750 | 11,250 | 11,028 | 10,000 | 11,250 | 22,431 | 62,500 | 50,000 | 87,500 | 100,000 | 106,250 |
| 39 | 0 | 137 | 177 | 479 | 316 | 507 | 92 | 2707 | 4227 | 12,272 | 750 | 6262 | 10,960 | 15,474 | 9346 | 37,501 | 68,778 | 65,193 | 55,165 | 132,178 |
| 40 | 0 | 3 | 0 | 0 | 0 | 13 | 0 | 0 | 0 | 16 | 1 | 1 | 2 | 11 | 0 | 0 | 0 | 0 | 0 | 3 |
| 41 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 42 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 43 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 44 | 0 | 40 | 61 | 193 | 80 | 103 | 101 | 985 | 2932 | 4821 | 2537 | 1850 | 3931 | 5359 | 376 | 13,541 | 23,843 | 25,267 | 23,124 | 46,487 |
| 45 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 46 | 31 | 158 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 4756 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 47 | 31 | 0 | 164 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2056 | 0 | 6163 | 8498 | 0 | 0 | 0 | 0 | 0 |
| 48 | 68 | 144 | 2 | 0 | 0 | 0 | 153 | 0 | 0 | 0 | 1746 | 1478 | 4 | 3 | 9941 | 1 | 0 | 0 | 1 | 0 |
| 49 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 50 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Sum | 466 | 1684 | 950 | 1501 | 1044 | 1838 | 2225 | 17484 | 29,210 | 55,548 | 49,271 | 46,219 | 52,092 | 61,699 | 110,478 | 304,712 | 274,218 | 431,786 | 488,853 | 568,953 |

**Table 7**
Comparison of the batch and proposed methods in the 10G TFT-LCD industry.

| Order # | # Of glass substrates | | Saving | |
|---|---|---|---|---|
| | Batch method (B) | Proposed method (P) | Quantities (B-P) | Ratio% (1-P/B) |
| 1 | 527 | 466 | 61 | 11.57 |
| 2 | 1901 | 1684 | 217 | 11.42 |
| 3 | 1094 | 950 | 144 | 13.16 |
| 4 | 1716 | 1501 | 215 | 12.53 |
| 5 | 1205 | 1044 | 161 | 13.36 |
| 6 | 2127 | 1838 | 289 | 13.59 |
| 7 | 2587 | 2225 | 362 | 13.99 |
| 8 | 19,962 | 17,484 | 2478 | 12.41 |
| 9 | 33,094 | 29,210 | 3884 | 11.74 |
| 10 | 63,296 | 55,548 | 7748 | 12.24 |
| 11 | 56,373 | 49,271 | 7102 | 12.60 |
| 12 | 53,550 | 46,219 | 7331 | 13.69 |
| 13 | 59,658 | 52,092 | 7566 | 12.68 |
| 14 | 70,612 | 61,699 | 8913 | 12.62 |
| 15 | 125,808 | 110,478 | 15,330 | 12.19 |
| 16 | 346,737 | 304,712 | 42,025 | 12.12 |
| 17 | 313,831 | 274,218 | 39,613 | 12.62 |
| 18 | 495,319 | 431,786 | 63,533 | 12.83 |
| 19 | 560,093 | 488,853 | 71,240 | 12.72 |
| 20 | 651,334 | 568,953 | 82,381 | 12.65 |

of TFT-LCD plates $i$ for $i = 1, \ldots, 9$ are $(f_1^1, f_1^2, f_1^3, f_1^4, f_1^5, f_1^6, f_1^7, f_1^8, f_1^9) = (0, 0, 0, 0, 0, 0, 0, 0, 6)$. The total number of plates is 6, the fitness value approximates 71%, and the algorithm finds 50 distinct cutting patterns within 55 s.

Additionally, the solution of pattern #20 (i.e., $g = 20$) in Table 4 is depicted in Fig. 7. The pattern contains 39 TFT-LCD plates, such that the numbers of plates $i$ for $i = 1, \ldots, 9$ are $(28, 2, 2, 0, 2, 1, 1, 1, 2)$. The coordinates of the pattern are listed in Table 5.

Details of the PP model for the 20 orders and the sets of feasible cutting patterns are shown in Table 6. Take order #1 as an example. The objective value is 466; and the solutions are $u_2 = 231$, $u_5 = u_6 = u_9 = 1$, $u_{22} = 2$, $u_{36} = 11$, $u_{37} = 20$, $u_{38} = 69$, $u_{46} = u_{47} = 31$, and $u_{48} = 68$. Thus, the proposed algorithm used 466 glass substrates to fulfill the first order.

In the Table 7, we compare the performance of the batch method with that of the proposed method. For order #1, the batch method and the proposed method require 527 and 466 glass substrates, respectively, to fulfill the first order. The proposed method saves 61 (11.57%) glass substrates. Moreover, the computation time of the proposed algorithm is approximately 60 s for each order. The results demonstrate that the proposed algorithm can effectively solve the CSPs in the 10G TFT-LCD industry.

## 5. Conclusion

This study proposes an integrated algorithm to solve the two-dimensional CSP in Taiwan's 10G TFT-LCD industry. The algorithm explores a GA, a CA method, and a PP method to solve CSPs efficiently. Compared with the current method, the proposed method can solve the same problem at a larger scale. On the other hand, to obtain a high-quality solution within a reasonable time, merging the column generation techniques, distributed algorithms, cloud computing, or other heuristic methods (i.e., neural network techniques, simulated annealing, and tabu-search) in the convergency of the CA method is a efficiency direction to enhance computational efficiency in future research.

## Acknowledgements

The authors would like to thank the editor and anonymous referees for providing most valuable comments for us to improve the quality of this manuscript. This research was supported by the project granted by ROC NSC 101-2811-E-009-011- and NSC 99-2221-E-030-005-.

## References

Beasley, J. E. (1985). An algorithm for the two-dimensional assortment problem. *European Journal of Operational Research, 19*, 253–261.
Chambers, M. L., & Dyson, R. G. (1976). The cutting stock problem in the flat glass industry selection of stock sizes. *Operational Research Quarterly, 27*, 949–957.
Chen, C. S., Lee, S. M., & Shen, Q. S. (1995). An analytical model for the container loading problem. *European Journal of Operational Research, 80*, 68–76.
Chen, C. S., Sarin, S., & Balasubramanian, R. (1993). A mixed-integer programming model for a class of assortment problems. *European Journal of Operations Research, 63*, 362–367.
Correia, M. H., Oliveira, J. F., & Ferreira, J. S. (2004). Reel and sheet cutting at a paper mill. *Computers & Operations Research, 21*, 1223–1243.
Cui, Y., & Lu, Y. (2009). Heuristic algorithm for a cutting stock problem in the steel bridge construction. *Computers & Operations Research, 36*(2), 612–622.
Dantzig, G. B. (1963). *Linear Programming and Extensions*. Princeton, NJ: Princeton University Press.
De Queiroz, T. A., Miyazawa, F. K., Wakabayashi, Y., & Xavier, E. C. (2012). Algorithms for 3D guillotine cutting problems: Unbounded knapsack, cutting stock and strip packing. *Computers & Operations Research, 39*(2), 200–212.
Demir, M. C. (2008). A pattern generation-integer programming based formulation for the carpet loading problem. *Computers & Industrial Engineering, 54*, 110–117.
Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of Np-completeness*. San Francisco, CA, USA: W.H. Freeman and Company.
Gonçalves, J. F. (2007). A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem. *European Journal of Operational Research, 183*(3), 1212–1229.
Gradisar, M., & Trkman, P. (2005). A combined approach to the solution to the general one-dimensional cutting stock problem. *Computers and Operations Research, 32*, 1793–1807.
Hinxman, A. I. (1980). The trim-loss and assortment problem: A survey. *European Journal of Operational Research, 5*, 8–18.
Holthaus, O. (2002). Decomposition approaches for solving the integer one dimensional cutting stock problem with different types of standard lengths. *European Journal of Operational Research, 141*, 295–312.
Jakobs, S. (1996). On genetic algorithms for the packing of polygons. *European Journal of Operational Research, 88*, 165–181.
Kroger, B. (1995). Guillotineable bin packing: A genetic approach. *European Journal of Operational Research, 84*(3), 645–661.
Lai, K. K., & Chan, W. M. (1997). An evolutionary algorithm for the rectangular cutting stock problem. *International Journal of Industrial Engineering, 4*(2), 130–139.
Land, A. H., & Doing, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica, 28*(3), 497–520.
Leung, T. W., Chan, C. K., & Troutt, M. D. (2003). Application of a mixed simulated annealing-genetic algorithm for a 2D packing problem. *European Journal of Operational Research, 145*, 530–542.
Li, H. L., & Chang, C. T. (1998). An approximately global optimization method for assortment problems. *European Journal of Operational Research, 105*, 604–612.
Li, H. L., Chang, C. T., & Tsai, J. F. (2002). Approximately global optimization for assortment problems using piecewise linearization techniques. *European Journal of Operational Research, 140*, 584–589.
Li, L., Ma, Y., Xu, N., Want, Y., & Hong, X. (2009). Modern floorplanning with boundary clustering constraint. In *IEEE computer society annual symposium on VLSI* (pp. 79–84).
Li, H. L., Tsai, J. F., & Hu, N. Z. (2003). A distributed global optimization method for packing problems. *Journal of the Operational Research Society, 54*, 419–425.
Lin, C. C. (2006). A genetic algorithm for solving the two-dimensional assortment problem. *Computers and Industrial Engineering, 50*, 175–184.
Page, E. (1975). A note on a two-dimensional dynamic problem. *Operational Research Quarterly, 26*, 321–324.
Pisinger, D. (2002). Heuristics for the container loading problem. *European Journal of Operational Research, 141*(2), 382–392.
Reinders, M. P. (1992). Cutting stock optimization and integral production planning for centralized wood processing. *Mathematical and Computer Modelling, 16*(1), 37–55.
Tang, M., & Yao, X. (2007). A memetic algorithm for VLSI floorplanning. *IEEE Transactions on System, Man, and Cybernetics-Part B: Cybernetics, 37*(1), 62–69.
Tsai, J. F., Hsieh, P. L., & Huang, Y. H. (2009). An optimization algorithm for cutting stock problems in the TFT-LCD industry. *Computers & Industrial Engineering, 57*, 913–919.
Umetani, S., Yagiura, M., & Ibaraki, T. (2003). One-dimensional cutting stock problem to minimize the number of different patterns. *European Journal of Operational Research, 146*, 388–402.
Wagner, B. J. (1999). A genetic algorithm solution for one-dimensional bundled stock cutting. *European Journal of Operational Research, 117*(2), 368–381.
Wang, Z., Li, K. W., & Levy, J. K. (2008). A heuristic for the container loading problem: A tertiary-tree-based dynamic space decomposition approach. *European Journal of Operational Research, 191*(1), 86–99.
Zheng, W., Ren, P., Ge, P., Qiu, Y., & Liu, Z. (2011). Hybrid heuristic algorithm for two-dimensional steel coil cutting problem. *Computers & Industrial Engineering, 62*(3), 829–838.