*sustainability*

MDPI

*Article*

# Computing the Assembly Guidance for Maximizing Product Quality in the Virtual Assembly

**Chen-Kun Tsung [1], Tseng-Fung Ho [2,*], Hsuan-Yu Huang [3], Shu-Hui Yang [3], Po-Nien Tsou [3], Ming-Cheng Tsai [3] and Yi-Ping Huang [3]**

1 Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, No. 57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung 41170, Taiwan; ckt@ncut.edu.tw
2 Department of Industrial Engineering and Management, National Chin-Yi University of Technology, No. 57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung 41170, Taiwan
3 Mechanical and Mechatronics System Research Labs, Industrial Technology Research Institute, No. 195, Sec. 4, Chung Hsing Rd., Chutung, Hsinchu 31040, Taiwan; itriA50248@itri.org.tw (H.-Y.H.); itriA40458@itri.org.tw (S.-H.Y.); BernieTsou@itri.org.tw (P.-N.T.); GregTsai@itri.org.tw (M.-C.T.); yiping@itri.org.tw (Y.-P.H.)
* Correspondence: fung@ncut.edu.tw

Received: 7 May 2020; Accepted: 29 May 2020; Published: 8 June 2020


**Abstract:** Assembly is the final process of manufacturing, and a good assembly plan reduces the effect of the tolerance generated in the early stages by the tolerance elimination. In the current assembly lines, the assemblers pick up the workpieces and install them together by the assembly instructions. When the workpieces are oversize or undersize, the product can not be installed correctly. Therefore, the assembler considers the secondary processing to fix the tolerance and then installs them together again. The product could be installed, but the product quality may be reduced by the secondary process. So, we formulate the assembly process as a combinatorial optimization problem, named by the dimensional chain assembly (DCA) problem. Given some workpieces with the corresponding actual size, computing the assembly guidance is the goal of the DCA problem, and the product quality is applied to represent the solution quality. The assemblers follow the assembly guidance to install the products. We firstly prove that the DCA problem is NP-complete and collect the requirements of solving the DCA problem from the implementation perspective: the sustainability, the minimization of computation time, and the guarantee of product quality. We consider solution refinement and the solution property inheritance of the single-solution evolution approach to discover and refine the quality of the assembly guidance. Based on the above strategies, we propose the assembly guidance optimizer (AGO) based on the simulated annealing algorithm to compute the assembly guidance. From the simulation results, the AGO reaches all requirements of the DCA problem. The variance of the computation time and the solution quality is related to the problem scale linearly, so the computation time and the solution quality can be estimated by the problem scale. Moreover, increasing the search breadth is unnecessary for improving the solution quality. In summary, the proposed AGO satisfies with the necessaries of the sustainability, the minimization of computation time, and the guarantee of product quality for the requirements of the DCA, and it can be considered in the real-world applications.

**Keywords:** dimensional chain assembly; virtual assembly; simulated annealing; combinatorial optimization

## 1. Introduction

The automatic and intelligent manufacturing is the vision in Industry 4.0. The automation means that the manufacturing process can be started and finish without human's operation. The intelligence allows the machines work together to discover the appropriate actions based on the actual status. To realize the automation and the intelligence, the managers have to capture the status during the manufacturing process. The infrastructure in current smart factories are ready in increasing the manufacturing efficiency. For example, the data analysis tool for the web [1], context-based service [2] and the transmission quality guarantee [3] are proposed to enhance the manufacturing processes. We can use the infrastructure to increase the transparency of manufacturing.

Assembly is the final stage of manufacturing, and the assembly line manager aims addressing the problem related to 5V properties, and they are Volume, Velocity, Variety, Veracity, and Value. Volume: many workpieces must be installed; Velocity: the assembly guidance must be determined as soon as possible; Variety: the target products are variant; Veracity: the assembly guidance must specify the correct workpieces; Value: the product must fit the requirements of the product specification.

The product quality is the major concern for the managers. Outputting under-qualified products results in the waste of time and money. Fortunately, the assembly process is the last step of manufacturing, and the product quality can be used to improve. A product consists of some workpieces, and the workpieces may touch with each other. Each contact surface of the workpiece has a design size and an actual size, and they may be different. In the design phase, we can apply the tolerance analysis models to calculate the virtual product tolerance [4]. The design size is the ideal value, but the manufacturing process may generate the tolerance. That is the major reason of that the actual size may be different from the design size. So, we can use the tolerance elimination to enhance the product quality in the assembly phase.

Here is an example to show the tolerance elimination. Figure 1 is the sectional drawing of a product that consists of two parts: shell and inner module. The assembler receives three workpieces per part, and the size of each workpiece is listed in Table 1. The inner module is installed in the shell, so the assembler should take one inner module and one shell and then installs them together. There is a gap between the inner module and the shell as shown in Figure 1, and we call the gap as the final size. Supposing the final size of the feasible product is bounded by 0 and 0.4, we can use the final size to evaluate the feasibility of the given assembly guidance (AG). For example, after receiving the assembly guidance $AG_1 = \{(1,1),(2,2),(3,3)\}$, the assembler installs the shell number 1 and the inner module number 1, the shell number 2 and the inner module number 2, and the shell number 3 and the inner module number 3 together. The final sizes are 0.2, 0.3, and 0.3, respectively, and $AG_1$ is feasible. However, $AG_2 = \{(1,3),(2,2),(3,1)\}$ is infeasible. The final sizes are 0.7, 0.3, and $-0.2$ in $AG_2$, and the final sizes of the first and third products are not acceptable, so $AG_2$ is infeasible.

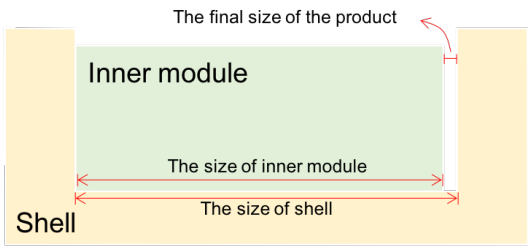



**Figure 1.** A instance of the product assembly with two parts, where the inner module should be installed in the shell.

**Table 1.** The size information of each workpiece in the example illustrated in Figure 1.

| Workpiece Index | Inner Module | Shell |
|---|---|---|
| 1 | 35.5 | 35.7 |
| 2 | 35.7 | 36 |
| 3 | 35.9 | 36.2 |

In the assembly line, the assembler receives some workpieces as shown in Figure 2 and the installation specification including the assembly instructions and the assembly process. The assembler randomly or sequentially picks up a workpiece of each part and then installs the workpiece one by one. Once the unacceptable final size is detected, the assembler will consider the secondary processing to fix the final size. However, the secondary processing is inappropriate, because the secondary processing only fixes a portion of size and lack for the dimensional chain evaluation [5,6]. So, the module's quality can not be guaranteed.

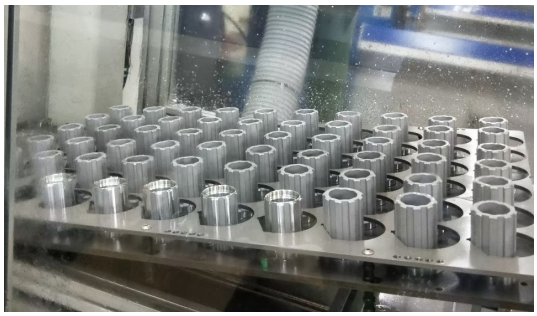


**Figure 2.** An example of the cassette body tray and some workpieces are waiting for cutting while some are finish.

An appropriate solution is to compute the AG in advanced, and the AG guides the assembler to install the products. Therefore, the properties of the products are controlled by the AG. To compute the AG, we have some considerations:

1. The sustainability: most production lines in the industry 4.0 factories provide non-stop works except for the maintenance. So, the AG computation must cover all kinds of product assembly processes that includes different number of workpieces and parts, the assembly sequence, the final size, etc.
2. The minimization of computation time: the AG computation can not disturb the assembly process. Once the assembly process is postponed because of the delay of the AG computation, the utilization of the assembly line is decreased. The AG computation can be processed while the workpieces are moving, and the AG can be prepared when the assemblers receive all workpieces.
3. The guarantee of product quality: when the assembler follows the assembly instruction provided by the AG, all output products are acceptable. In other words, the AG computation has to guarantee the product quality.

To reach the necessary listed above, we first formulate the AG computation as the dimensional chain assembly (DCA) problem, and the goal of DCA problem is to output the acceptable AG for assemblers. We prove that the DCA problem is a NP-complete problem, and the search algorithms [7,8] requires huge computation time to find the optimal solution. The requirement of the minimum computation time is violated. The combinatorial optimization techniques help us to compute the optimal solutions [9], but the combinatorial optimization algorithms may be modified for installing new products. So, we survey the general optimization approaches that are satisfied with the sustainability. We adopt the single-solution evolution (SSE) to reach the consideration of the minimum computation time. To guarantee of product quality, we modify the simulated annealing (SA) algorithm to design the assembly guidance optimizer (AGO) to calculate the AG.

To evaluate the performance of the proposed AGO, we build a Windows-based platform to simulate the AG computation, and measure the performance of the AGO. We obtain following properties from the simulated results:

1. The sustainability: the assemblers obtain the AG from the proposed AGO in the different DCA problems (to install various products). So, the assembly manager can use the AGO to compute the AG sustainably.
2. The minimization of computation time: the AGO outputs the AG for installing thousands of workpieces in few seconds, and the computation is finish before all workpieces arrive at the assembly line.
3. The guarantee of product quality: the final size of all products are satisfied with the product specification, and the AG outputted by the AGO provides high solution quality.

Moreover, given an assembly configuration of the AGO, the increase of the computation time and the decrease of the solution quality is linear to the problem scale. It means that the computation time and the product quality of the AGO can be predicted by the scale of the assembly instance. Therefore, the AGO can be applied to the real-world assembly lines for computing the AGs of various products.

## 2. Related Works

Calculating the optimal solution with minimum gap between the product size and the ideal size in the DCA problem can be reduced to the exact weight perfect matching problem from a given bipartite graph. Therefore, solving the DCA problem is NP hard, as shown in Section 3.3. There are two major approaches of solving the DCA problem, and they are listed as follows:

- Greedy algorithms: deriving the near-optimal or optimal (when the instances meet the specific condition) solutions by the problem properties [10,11].
- Soft-computing algorithms: deriving the near-optimal solutions by continuously refining the solution quality [9,12–15].

Greedy algorithms output the solutions efficiently, but the optimal solution is not guaranteed. Soft-computing algorithms seek the solutions with better quality one by one, so soft-computing algorithms require more computation time than that of greedy algorithms. Moreover, because greedy algorithms are problem-based approaches, the algorithms should be modified in solving different problems. Therefore, the soft-computing algorithms are more appropriate than greedy algorithms in terms of the industrial purpose.

Although the soft-computing algorithms require more computation time to derive near-optimal solutions, we still can get the acceptable solutions in a short period of time because of the increase of the computation power. Therefore, soft-computing algorithms are applied to broad-range applications, such as the decision evaluation in the banks [14,16], the timetable calculation of the train scheduler [17,18], the vehicle route determination [19,20], etc.

Soft-computing algorithms are classified into two categories: the single-solution evolution (SSE) [12] and the multiple-solution evolution (MSE) approaches, e.g., such as genetic algorithm [14,21], particle swarm optimization [22], and ant colony optimization [23]. In each iteration, the SSE approaches only consider one solution while some solutions are evaluated by the MSE approaches. The SSE approaches output the improved solutions rapidly [12,13], and the solution quality is continuously improved. On the other hand, MSE approaches require more computation time to finish an iteration, but MSE approaches have higher probability in obtaining better solutions than SSE. Therefore, SSE approaches are more appropriate than MSE approaches based on the industrial consideration, and we apply the SSE approach to design the AGO.

The local search approaches [24] and the simulated annealing (SA) [12,13] are popular SSE approaches. The local search approaches consider problem properties to search the solutions with higher quality. SA applies the annealing idea to approximate optimal solution iteratively for the

discrete solution space. SA covers broader range applications than local search approaches. Therefore, SA is more appropriate than local search approaches because the assemblers have to install various products in the assembly lines.

## 3. Preliminaries

### *3.1. Problem Definition*

**Definition 1.** *Given a DCA problem $D = (WP, EQ, SPEC)$ consisting of the workpiece information $WP$, assembly equation $EQ$, and assembly specification $SPEC$. The goal of DCA problem is to find an assembly guidance $AG$ in the specific time consumption $T_r$.*

Considering $m$ parts and $n$ workpieces for each part, we have $mn$ workpieces in total in the DCA problem. To install one product, the assembler picks up one workpiece per part and installs them together, and the assembly line eventually generates $n$ products. In this paper, the goal is to design a selection algorithm to compute $AG$ to help assemblers to install products.

The DCA problem $D = (WP, EQ, SPEC)$ has following components:

- $WP = \{wp_{ij}\}, \forall i \leq m$ and $j \leq n$, is the set of the size of all workpieces. For the workpiece $j$ of the part $i$, we use the term $wp_{ij}$ to represent the actual size of the workpiece. In the design phase, the engineer computes the design sizes and the tolerances. However, the actual size may be different with the design size because of the manufacturing process.
- $EQ = \{eq_1, eq_2, \ldots, eq_m\}, \forall eq_i \in \{1, -1\}$ is the assembly equation. The workpiece with $eq_i = -1$ should be installed in the workpiece with $eq_j = 1$. The engineers prepare $EQ$ in the design phase, so $EQ$ is ready for computing the $AG$ in the Manufacturing Execution System (MES) after the design phase.
- $SPEC = (cv, pt, nt)$ is the product specification which includes a central value $cv$, a positive tolerance $pt$, and the negative tolerance $nt$, where $cv > 0$, $pt \geq 0$, and $nt \leq 0$ to indicate the feasible product size. Each product has a final size after the assembly process. The final size is controlled within a range by $pt$ and $nt$ for the functionality consideration. Thus, the final size can be used to measure the product quality. When the final size of $k$-th product $gap^k$ is satisfied with $(cv + nt) \leq gap^k \leq (cv + pt)$, the product is acceptable, and failed otherwise.

The solution of the DCA problem is $AG$ that indicates the workpiece installation information of each product. So, the assemblers follow the information listed in $AG$ to pick up the corresponding workpieces and install them together. The production quality can be evaluated in advanced. Here, we list some solution definition:

- The assembly guidance $AG = \{G^1, G^2, \ldots, G^n\}$: $G^k$ denotes the assembly indication of each workpiece for $k$-th product, $\forall k \leq n$.
- The indicated variable $x_{ij}^k = \{0, 1\}$: the workpiece $ij$ is selected in the product $k$ when $x_{ij}^k = 1$, and not considered for $x_{ij}^k = 0$. Thus, we have $G^k = \{x_{ij}^k\}, \forall i \leq m$ and $\forall j \leq n$.
- The product final size $gap^k$: we measure $gap^k$ for the product $k$ using the workpieces listed in $G^k$, and we have:

$$gap^k = \sum_{i=0}^{m} \sum_{j=0}^{n} eq_i \times wp_{ij} \times x_{ij}^k. \tag{1}$$

- The product quality $(gap^k - cv)$: we use the largest gap between $gap^k$ and $cv$ to be the solution quality of $AG$, and we have:

$$\arg\max_{\forall k} \left| gap^k - cv \right|. \tag{2}$$

- The acceptable product quality $(cv + nt) \leq gap^k \leq (cv + pt)$: in other words, we have:

$$nt \leq (gap^k - cv) \leq pt. \tag{3}$$

Therefore, the objective function of the DCA is to minimize the value between $gap^k$ and $cv$ for each product $k$.

$$\min_{\forall k} \left| (gap^k - cv) \right|. \tag{4}$$

For the convenience, we use the notation $q(sol)$ to identify the solution quality of $sol$, i.e., $q(sol) = \max_{\forall k} \left| (gap^k - cv) \right|$. So, we can use $q(sol)$ to trace the improvement of the solution quality. Moreover, some constraints of a feasible solution are listed as follows:

1. $\sum_{j=0}^{n} x_{ij}^k = 1, \forall 0 \leq i \leq m$: the product $k$ includes one workpiece per part.
2. $\sum_{k=0}^{n} x_{ij}^k = 1, \forall 0 \leq i \leq m$ and $\forall 0 \leq k \leq n$: the workpiece $ij$ appears in exactly one product.
3. $\sum_{i=0}^{m} \sum_{j=0}^{n} x_{ij}^k = m$: the number of workpieces in each product is $m$.

### 3.2. A Case Study

We use the example illustrated in Figure 1 to show the DCA problem. The assembler picks up a workpiece from shell and another one from inner module, and then puts the inner module in the shell. Considering $m = 2$ and $n = 3$, we have six workpieces. The size information is shown in Table 1, and that is $WP$. Because the inner module is installed in the shell, we have $EQ = \{-1, 1\}$. Supposing $SPEC = (0.2, 0.2, -0.2)$, the final size of an acceptable product is between 0 and 0.4. Considering two solutions listed in Table 2, solution 1 is an acceptable because the final product sizes are all within 0 and 0.4. However, solution 2 is unacceptable because the final size of the first product is 0.7 which is out of the specified size range.

**Table 2.** Two solutions of the example shown in Table 1. For convenience, we use label A to identify the shell while B for the inner module. For the first product in solution 1, 1A + 1B means that the product number one is installed by first shell and first inner module, and the final size is 0.2.

| Product Index | Solution 1 | | Solution 2 | |
|---|---|---|---|---|
| | **Assembly** | **Final Size** | **Assembly** | **Final Size** |
| 1 | 1A + 1B | $35.7 - 35.5 = 0.2$ | 1A + 3B | $36.2 - 35.5 = 0.7$ |
| 2 | 2A + 2B | $36 - 35.7 = 0.3$ | 2A + 1B | $35.7 - 35.7 = 0$ |
| 3 | 3A + 3B | $36.3 - 35.9 = 0.4$ | 3A + 2B | $36 - 35.9 = 0.1$ |

### 3.3. Computational Complexity Analysis

**Theorem 1.** *Given a DCA problem $D = (WP, EQ, SPEC)$, the goal of $D$ is to compute an assembly guidance $AG$ that each product fits the requirement specified in $SPEC = (cv, pt, nt)$.*

**Proof.** To prove that the DCA problem is NP-complete, we reduce the subset of the DCA from the exact weight perfect matching problem [25]. Consider a DCA problem $D_2 = (WP_2, EQ_2, SPEC_2)$ that each product consists of two workpieces, e.g., $m = 2$, as shown in Figure 1. We can illustrate $D_2$ as a bipartite graph that the left vertices in the bipartite graph includes part one workpieces while part two workpieces are listed in the right-hand side. Each edge $e_{jj'}$ represents the connection from $wp_{1j}$ to $wp_{2j'}$ and the weight $w_{jj'}$ is the size of the final product. Moreover, $e_{jj'}$ exists only if each value of $w_{jj'}$ is satisfied with the specification listed in $SPEC$. So, the goal is to calculate a perfect matching from the bipartite graph.

According to the reduction of Zhu et al. [25], the exact weight perfect matching of bipartite graph is NP-complete, and $D_2$ is also NP-complete. $D_2$ considers the products with two workpieces, and we have $D_2 \subseteq D$. Therefore, $D$ is NP-complete because of $D_2 \subseteq D$. □

## 4. Proposed Solution

According to the survey of feasible approaches listed in Section 2, we apply the SA algorithm to design the AGO for computing the assembly guidance with $\min_{\forall k} \left|(gap^k - cv)\right|$. The algorithm of AGO is shown in Algorithm 1. The AGO receives a DCA instance $D$, an initial temperature $T$, a temperature descent rate $r$, and a maximum number of iterations $iter$. The AGO outputs an assembly guidance $sol$. Firstly, the AGO generates an initial solution $sol$ in line 1. Then, the AGO enters a solution refinement loop for seeking better solutions from lines 2 to 8. When meeting the stop conditions, the AGO exits the loop and return the refined solution $sol$. In the refinement loop, the AGO picks up a neighbor solution $sol_{nb}$ based on $sol$ and then evaluates the acceptance of $sol_{nb}$ by the quality of $sol_{nb}$ and the current temperature. In the last step of the solution refinement loop, the temperature is reduced by $r$. Here, we show the details about the AGO algorithms.

**Algorithm 1:** The algorithm of the proposed assembly guidance optimizer (AGO) for DCA.

**input :** DCA instance $D$, initial temperature $T$, temperature descent rate $r$, maximum iteration $iter$, variation degree $v$
**output:** The assembly guidance $sol$

```
1 generate an initial solution sol ;
2 while not meet the stop condition do
3     find a neighbor solution sol_nb ← nbFinder(sol, v);
4     if ΔE > 0 then
5         sol ← sol_nb;
6     else if e^(ΔE/T) > random(0,1) then
7         sol ← sol_nb;
8     T ← T * r;
9 return sol ;
```

- The solution encoding: the solution structure is defined as $AG$ in Section 3.1. However, $AG$ is sparse because only $m$ elements are meaningful for each product. Therefore, we design another data structure for reducing the memory usage and increasing the computation efficiency for the implementation consideration. Two solutions listed in Table 2 are transformed to the format as shown in Table 3. The column indicates the part information while each row represents the selected workpieces in a product. Each element in the solution is the workpiece index. So, $sol$ provides higher readability for the assembler. In the solution 2, for example, the first product considers workpiece 1 of part A and workpiece 3 of part B, the second product considers workpiece 2 of part A and workpiece 1 of part B, etc. Moreover, the format of $sol$ is still satisfied with the constraints listed in Section 3.1.
- The initial solution generation: the AGO apply the random process to construct the initial solution. The AGO is designed for installing various products. Because the assembly processes and the properties of various products are different, considering the random initial solution increases the coverage of the search directions. For an initial solution $sol$, each workpiece is picked up randomly without putting back and inserted into an arbitrary product.

- The neighbor solution generation: we refer to the "royal road" function [26] to design a neighborhood search approach. The algorithm named by $nbFinder()$ is illustrated in Algorithm 2. $nbFinder()$ receives two parameters: the base solution $sol$ and the variation degree $v$. $nbFinder()$ outputs a solution $sol_{nb}$ based on $sol$, and $v$ determines the difference between $sol$ and $sol_{nb}$. Firstly, $nbFinder()$ copies $sol$ to $sol_{nb}$. Then, $nbFinder()$ picks up one part and two products as shown in lines 3 to 5. Then, the workpieces of the selected products are swapped as illustrated in Figure 3. Iteratively executing the exchange process for $v$ times. Therefore, $v$ controls the variation degree of $sol$ and $sol_{nb}$.

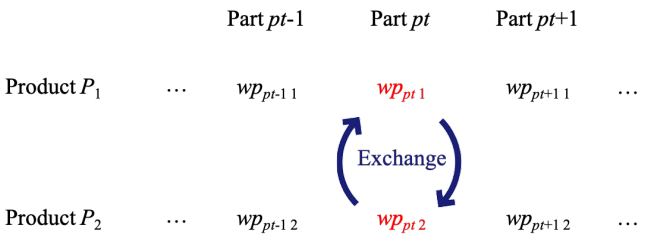



**Figure 3.** An example of generating the neighbor solution. Given a solution $sol$, the process randomly selects two products and a part, e.g., the products $P_1$ and $P_2$ and the part $pt$, and then exchanges the workpieces $wp_{pt1}$ and $wp_{pt2}$ to generate the neighbor solution $sol_{nb}$.

- The stop condition: the AGO considers the maximum iteration $iter$ and the minimum temperature as the stop condition. The value of $iter$ given by the assembler manager controls the running time of the AGO. According to the annealing concept, the temperature is continuously reduced to the room temperature, so the room temperature is also another stop condition. By considering the linear temperature reducing function, we assume the room temperature is 0 °C. Therefore, the temperature could be very closed to zero but not smaller than zero. The maximum number of iterations or the minimum temperature will not dominate the stop condition, and the temperature reducing function can work together with the maximum number of iterations.
- The acceptance criterions of the neighbor solutions: the AGO computes the $\Delta E$ by the difference between the solution quality between $sol_{nb}$ and $sol$, i.e., $\Delta E = q(sol_{nb}) - q(sol)$. For $\Delta E < 0$, i.e., $q(sol_{nb}) < q(sol)$, the quality of $sol_{nb}$ is better than that of $sol$, and the AGO accepts $sol_{nb}$. On the other hand, the AGO simulates the Boltzmann distribution [27] to determine the probability of accepting $sol_{nb}$. According to the property of the Boltzmann distribution, $sol_{nb}$ with lower quality is accepted easily in the early stage. The AGO is designed to discover as wide as possible, so the probability of accepting the solution with worse quality in the early stage is higher than that in the late stage. The neighbor solution acceptance concept is listed in lines 4 to 7 in Algorithm 1.

**Table 3.** The solution example of Table 2 considered in the AGO.

| Product Index | Solution 1 | | Solution 2 | |
|---|---|---|---|---|
| | Part A | Part B | Part A | Part B |
| 1 | 1 | 1 | 1 | 3 |
| 2 | 2 | 2 | 2 | 1 |
| 3 | 3 | 3 | 3 | 2 |

**Algorithm 2:** ($nbFinder(sol, v)$) The algorithm of the neighbor solution generation.

**input** : a solution $sol$, variation degree $v$
**output**: a neighbor solution $sol_{nb}$

```
copy(sol, sol_nb);
for y ← 1 to v do
    compute pt randomly from 1 to m;
    compute p_1 randomly from 1 to n;
    compute p_2 randomly from 1 to n s.t. p_2 ≠ p_1;
    swap(sol_nb, pt, p_1, p_2);
return sol_nb ;
```

## 5. Simulation

The AGO provides high flexibility for the assembly processes of various products. The assembly manager has to determine the AGO configurations to compute the assembly guidance. Therefore, we first compute the optimal configuration before evaluating the performance of the AGO. Next, we apply the derived configuration to evaluate the AGO performance including the computation efficiency and the solution quality.

We consider the countershaft module as shown in Figure 4 to be the simulated target [28]. In this case, each product consists of 11 parts, i.e., $m = 11$. The letter from A to L in Figure 4 represents a cutting plane, e.g., AB, BC, CD, DE, EF, FG, GH, HI, IJ, JK, and KL. All workpieces are installed in the central columella, so all cutting planes are in the same dimensional chain. Therefore, we have $EQ = \{-1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1\}$ for the sequence of AB, BC, CD, DE, EF, FG, GH, HI, IJ, JK, and KL. We consider $cv = 0.3$, $pt = 0.15$, and $nt = -0.15$. Given $n$ workpieces for each part, the assemblers receive $11n$ workpieces in total, and output $n$ countershaft modules.

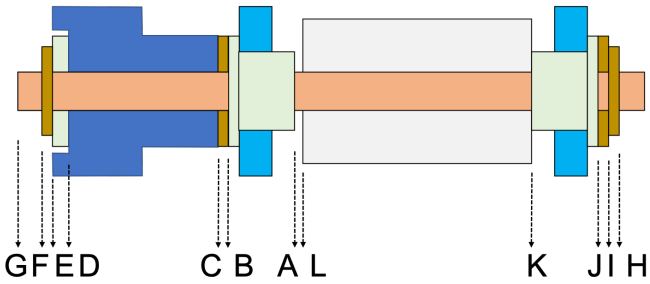



**Figure 4.** An example of the assembly guide of the countershaft module.

We consider the personal computer as the simulation platform with Windows 10. The platform equips Intel i7 CPU, 16 GB memory, and 512 GB SSD while the AGO is implemented by C# in Visual Studio 2015.

### *5.1. Configuration Evaluation*

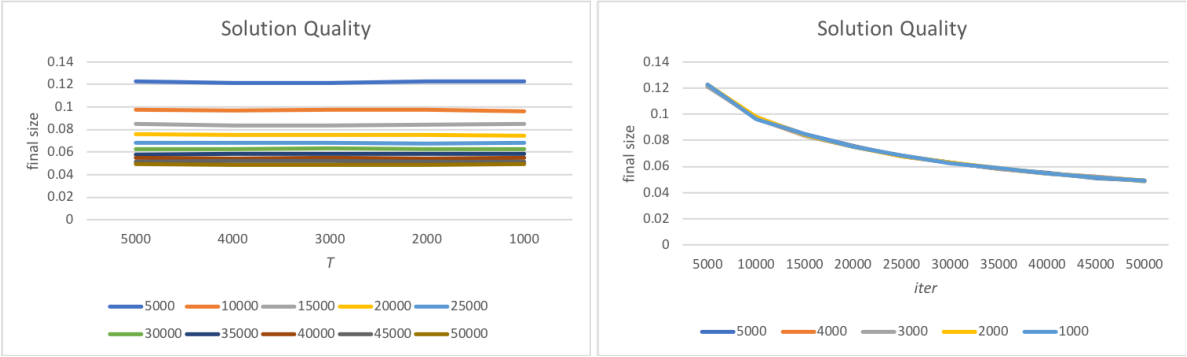
We consider 1500 workpieces for each part, e.g., $n = 1500$. The AGO configuration includes $T$, $r$, $iter$, and $v$. We focus on evaluating the settings of $T$, $r$, and $iter$, and $v = 1$ is applied to the configuration evaluation. The considered configurations are: $T$ from 5000 to 50,000 with gap 5000, $r$ from 0.9 to 0.99 with gap 0.01, and $iter$ from 1000 to 5000 with gap 1000. For each configuration, we run 10 times and use the average values to be the evaluation results. We consider the average CPU time and the average solution quality as defined in Equation (2) to measure the configuration quality.

#### 5.1.1. Solution Quality Evaluation

The final size of each product is the major concern for the assembly manager. We first investigate the solution quality for all parameter combinations to find out the appropriate configurations that will be applied to the next-step simulations. Figure 5 lists the solution quality results defined in Equation (2) for all combinations with *iter* and $T$ under $v = 0.98$. We have following observations:

1. The distribution of the solution is similar for various settings of $r$. The results in Figure 5 is captured in the configurations with $r = 0.98$, and the distribution is similar to that captured in other settings of $r$. So, we just illustrated the results with $r = 0.98$.
2. The difference of the solution quality between various settings of $T$ is small from Figure 5b, and the solution quality is not improved dramatically by increasing the initial temperature.
3. Higher settings of *iter* lead to better solution quality in Figure 5a because the solutions can be refined for longer search time.

From the above observations, the settings of $T$ and *iter* are not the critical parameter, so we evaluate the setting of $r$. We capture the best solution from all settings of $r$, and the results are listed in Figure 6 and Table 4. The result shows that the setting $r = 0.98$ provides minimum final size. Therefore, we will consider $r = 0.98$ in the following simulations.




(**a**) The solution quality comparison in $T$. (**b**) The solution quality comparison in *iter*.

**Figure 5.** The solution quality results defined by Equation (2) for all combinations with *iter* and $T$ where the setting of $v$ is 0.98.

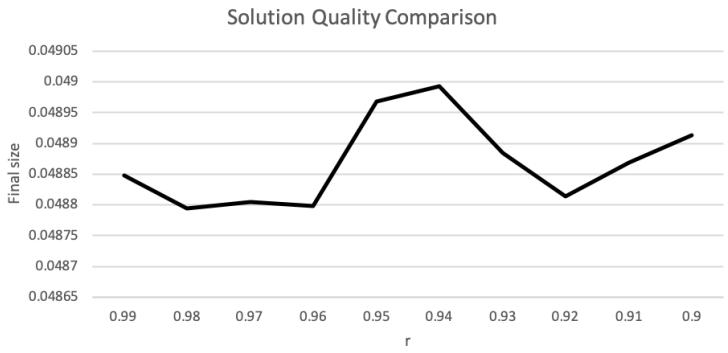



**Figure 6.** The evaluation results captured from various $r$.

**Table 4.** The best solution quality is captured in all combinations of *iter* and $T$ for the settings of $r$.

| *r* | 0.99 | 0.98 | 0.97 | 0.96 | 0.95 | 0.94 | 0.93 | 0.92 | 0.91 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Final size | 0.04885 | 0.04879 | 0.0488 | 0.04879 | 0.04896 | 0.04899 | 0.04888 | 0.04881 | 0.04886 | 0.04891 |

#### 5.1.2. Marginal Improvement Evaluation

From the results listed in Figure 5, the solution quality is improved by increasing the number of iterations. Therefore, we compare the marginal improvement for the settings of $iter$ and $T$. The marginal improvement is defined as:

$$mi(iter, T) = \frac{\Delta finalGap}{\Delta runnignTime}. \quad (5)$$

Given the settings of $iter$ and $T$ with $r = 0.98$, $\Delta finalGap$ stands for the difference of the final gap between $iter$ and $(iter + 5000)$ while $\Delta runningTime$ represents the gap of the running time. Equation (5) shows the improvement ratio of the solution quality to the running time. Therefore, maximizing the values of $mi(iter, T)$ is the objective in this simulation.

We have the results illustrated in Figure 7. The $mi(iter, T)$ curves are descending for all configurations as increasing the settings of $iter$. However, the maximum $mi(iter, T)$ takes place from the configurations with $iter = 10{,}000$ and $T = 5000$. The result is reasonable, so we will use this configuration to evaluate the performance of the proposed AGO.

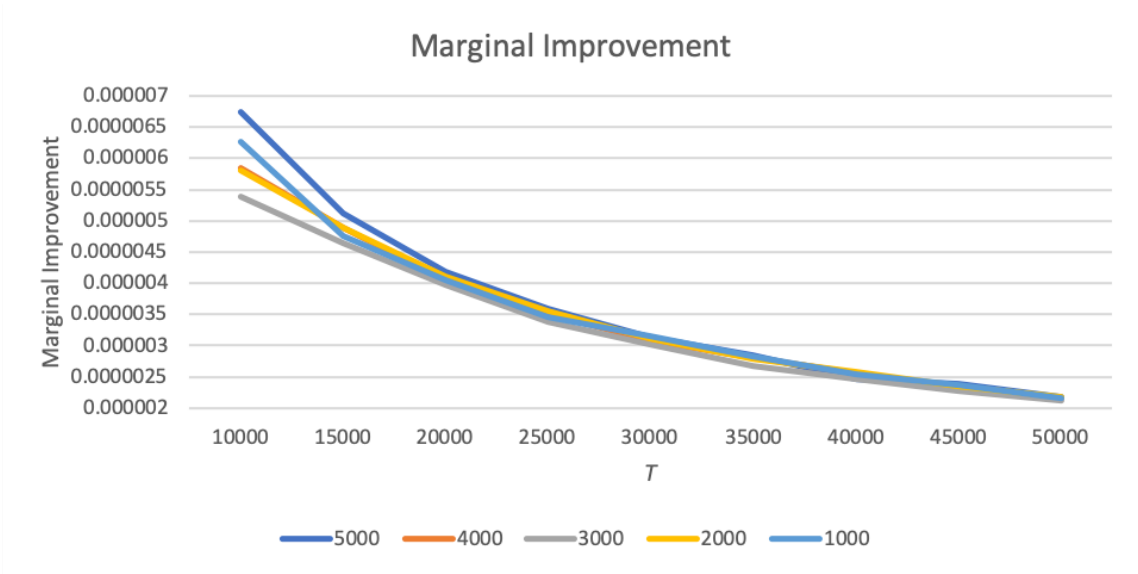


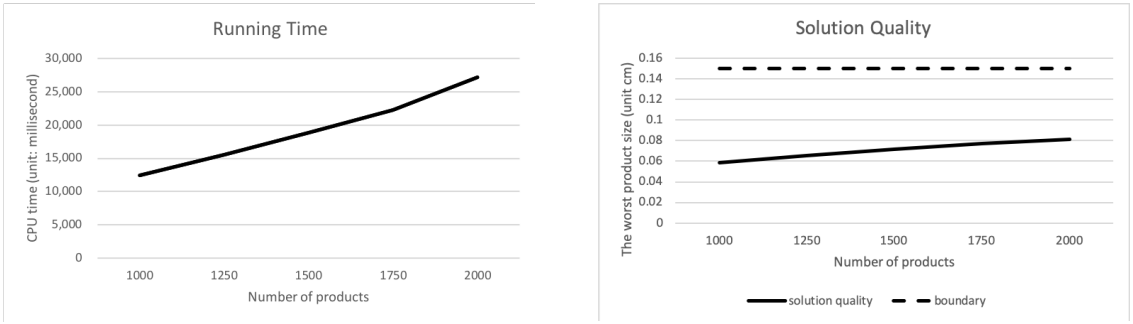

**Figure 7.** The marginal improvement evaluation of the AGO.

### *5.2. Performance Evaluation*

We consider $T = 5000$, $r = 0.98$, and $iter = 10{,}000$ in the following experiments for measuring the AGO performance. We generate 10 instances for $n = 1000, 1250, 1500, 1750$, and 2000. We run the AGO 10 times for each instance and use the averaged value for each configuration. The results in terms of the running time and the solution quality are illustrated in Figure 8.

The curve of running time is linearly raised by the number of products. The AGO requires more computation time in each iteration of larger scale instances. So, this is the major reason for the raised computation time even if the AGO parameters are the same. The AGO uses 25 seconds approximately to compute the assembly guidance in the instances with $2000 \times 11$ workpieces. The running time is acceptable for the implementation consideration.

The gap between the product size and the ideal size is getting bigger for large scale instances. The AGO requires more computation time to maintain the same solution quality in the large-scale instances. The size of the search space is increased exponentially by the instance scale, e.g., the number of workpieces, so the solution quality is decreased by the number of the products in the same AGO configuration. The averaged gap between final size and $cv$ is slightly increased from 0.0589 to 0.0816 without providing more computation resource.



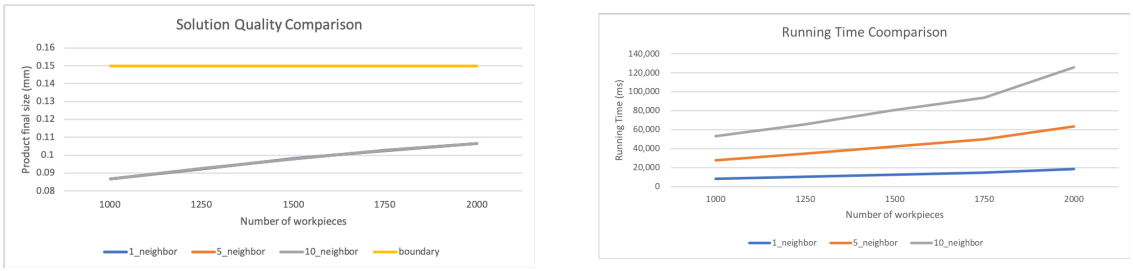

(**a**) (**b**)

**Figure 8.** The simulation results for the performance of the AGO. (**a**) The running time captured from the instances with various number of products. (**b**) The solution quality captured from the instances with various number of products, where the dot line indicates the upper bound of the acceptable product size, i.e., *cv*.

The running time is increased and the solution quality is decreased by scaling up the problem size, but the variance amount is linear. It means that the assembly manager can estimate the running time and the solution quality from the problem scale, and the configuration is unnecessary to be re-evaluated. The configuration of AGO should be re-evaluated only when the solution quality is very close to the target *cv* or over *cv*, where the products are near unacceptable.

### *5.3. Search Breadth Evaluation*

Single solution refinement is the major property of the SA, and the neighbor solutions can be carefully evaluated rather than the wide search. However, the solution quality is difficult to be improved dramatically in the SA. We are interested in the effect of the search breadth, so we estimate the performance of the AGO with different degrees of the search breadth. There are two ways to increase the search breadth: increasing the number of the population size, and increasing the degree of the local search. Since the SA is an SSE approach, we do not increase the number of the population size. Thus, we evaluate some neighbors in the local search approach to increase search breadth. We consider 1, 5, and 10 neighbors in each experiment and evaluate the running time, the solution quality, and the computation efficiency for the configuration with $r = 0.98$, $T = 5000$, and $iter = 10{,}000$. The solution results are illustrated in Figure 9.




(**a**) (**b**)

**Figure 9.** The experiment results about the running time and the solution quality. (**a**) The solution quality captured from the instances with various number of products, where the dot line indicates the upper bound of the acceptable product size. (**b**) The running time captured from the instances with various number of products.

From the results in Figure 9a, the solution quality is not improved by increasing the number of evaluated neighbors. On the other hand, it is rational to receive that the running time is increased in high number of neighbors as shown in Figure 9b. Increasing the search breadth is profitless in improving the solution quality. Therefore, the AGO with the single neighbor evaluation process is efficient and outputs high-quality solution.

## 6. Conclusions

In this paper, we define the DCA problem and prove the problem is NP-complete by the reduction from the exact weight perfect matching problem. We propose a SA-based approach that is named as AGO to resolve the DCA problem. The AGO considers 5V properties of the assembly process. The simulation results show that the problem scale leads to the linear effect on the solution quality and the running time. Therefore, the assembly manager can easily estimate the running time and the solution quality. Moreover, the AGO provides appropriate search strategy, and it is unnecessary to search multiple solutions simultaneously. So, the near optimal solution can be calculated efficiently.

There are some variance models of the DCA problem. For example, a target product may have several dimensional chains. The dimensional chains may cross on a special part, so the AGO has to make sure that the requirements of all dimensional chains can be satisfied in a target product. Based on the current progress, we have begun modeling the multi-dimensional chain problem, and we will apply the AGO to resolve the multi-dimensional chain problem where that is close to the real-world assembly goal.

**Author Contributions:** Data curation: H.-Y.H., S.-H.Y., P.-N.T., M.-C.T. and Y.-P.H.; Methodology: C.-K.T., T.-F.H., H.-Y.H., S.-H.Y., P.-N.T., M.-C.T. and Y.-P.H. Project administration: C.-K.T.; Software: C.-K.T., T.-F.H., H.-Y.H., S.-H.Y., P.-N.T., M.-C.T. and Y.-P.H.; Writing—original draft preparation: C.-K.T., T.-F.H., H.-Y.H., S.-H.Y., P.-N.T., M.-C.T. and Y.-P.H.; Writing—review and editing: C.-K.T., H.-Y.H., S.-H.Y., P.-N.T., M.-C.T. and Y.-P.H. All authors have read and agreed to the published version of the manuscript.

**Funding:** This study was supported in part by grants from the Ministry of Economic Affairs of the Republic of China (Grant No. J353CH1320) and the Ministry of Science and Technology of the Republic of China (Grant No. MOST 108-2218-E-167-003-MY2).

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| DCA | Dimensional Chain Assembly |
| SA | Simulated Annealing |
| AG | Assembly Guidance |
| AGO | Assembly Guidance Optimizer |
| MES | Manufacturing Execution System |

## References

1. Tseng, C.H.; Chen, Y.H.; Jiang, Y.R. The implementation of an automatic web-driven data analysis framework. *Int. J. Soc. Humanist. Comput.* **2017**, *2*, 150–165. [CrossRef]
2. Pallasena, R.K.; Sharma, M.; Krishnaswamy, V. Context-sensitive smart devices-definition and a functional taxonomy. *Int. J. Soc. Humanist. Comput.* **2019**, *3*, 108–134. [CrossRef]
3. Xu, X. The analytics and applications on supporting big data framework in wireless surveillance networks. *Int. J. Soc. Humanist. Comput.* **2017**, *2*, 141–149. [CrossRef]
4. Cao, Y.; Liu, T.; Yang, J. A comprehensive review of tolerance analysis models. *Int. J. Adv. Manuf. Technol.* **2018**, *97*, 3055–3085. [CrossRef]
5. Tsai, J.C.; ; Chen, F.C.; Dai, J.H. Reduction of tolerance stack-up by grouped random assembly for components with uniform distributions. *Procedia CIRP* **2015**, *27*, 260–263. [CrossRef]
6. Lin, C.Y.; Huang, W.H.; Jeng, M.C.; Doong, J.L. Study of an assembly tolerance allocation model based on Monte Carlo simulation. *J. Mater. Process. Technol.* **1997**, *70*, 9–16. [CrossRef]
7. Li, J.; Marier, D.; Tufte, K.; Papadimos, V.; Tucker, P.A. No pane, no gain: Efficient evaluation of sliding-window aggregates over data streams. *SIGMOD Recore* **2005**, *34*, 39–44. [CrossRef]
8. Bentley, J.L. Multidimensional binary search trees used for associative searching. *Commun. ACM* **1975**, *18*, 509–517. [CrossRef]

9. Blum, C.; Puchinger, J.; Raidl, G.R.; Roli, A. Hybrid metaheuristics in combinatorial optimization: A survey. *Appl. Soft Comput.* **2011**, *11*, 4135–4151. [CrossRef]
10. Bang-Jensen, J.; Gutin, G.; Yeo, A. When the greedy algorithm fails. *Discret. Optim.* **2004**, *1*, 121–127. [CrossRef]
11. Goyal, A.; Lu, W.; Lakshmanan, L.V. Celf++ optimizing the greedy algorithm for influence maximization in social networks. In Proceedings of the 20th International Conference Companion on World Wide Web, Hyderabad, India, 28 March–1 April 2011; pp. 47–48.
12. Wang, Y.; Bu, G.; Wang, Y.; Zhao, T.; Zhang, Z.; Zhu, Z. Application of a simulated annealing algorithm to design and optimize a pressure-swing distillation process. *Comput. Chem. Eng.* **2016**, *95*, 97–107. [CrossRef]
13. Zhan, S.H.; Lin, J.; Zhang, Z.J.; Zhong, Y.W. List-based simulated annealing algorithm for traveling salesman problem. *Comput. Intell. Neurosci.* **2016**, *2016*, 1712630. [CrossRef] [PubMed]
14. Metawa, N.; Hassan, M.K.; Elhoseny, M. Genetic algorithm based model for optimizing bank lending decisions. *Expert Syst. Appl.* **2017**, *80*, 75–82. [CrossRef]
15. Khalil, E.; Dai, H.; Zhang, Y.; Dilkina, B.; Song, L. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*; NIPS: Long Beach, CA, USA, 2017; pp. 6348–6358.
16. Soui, M.; Gasmi, I.; Smiti, S.; Ghédira, K. Rule-based credit risk assessment model using multi-objective evolutionary algorithms. *Expert Syst. Appl.* **2019**, *126*, 144–157. [CrossRef]
17. Caprara, A.; Monaci, M.; Toth, P.; Guida, P.L. Lagrangian heuristic algorithm for a real-world train timetabling problem. *Discret. Appl. Math.* **2006**, *154*, 738–753. [CrossRef]
18. Liu, L.; Dessouky, M. A decomposition based hybrid heuristic algorithm for the joint passenger and freight train scheduling problem. *Comput. Oper. Res.* **2017**, *87*, 165–182. [CrossRef]
19. Sitek, P.; Wikarek, J. Capacitated vehicle routing problem with pick-up and alternative delivery (CVRPPAD): Model and implementation using hybrid approach. *Ann. Oper. Res.* **2019**, *273*, 257–277. [CrossRef]
20. Chagas, J.B.; Silveira, U.E.; Santos, A.G.; Souza, M.J. A variable neighborhood search heuristic algorithm for the double vehicle routing problem with multiple stacks. *Int. Trans. Oper. Res.* **2020**, *27*, 112–137. [CrossRef]
21. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T.A.M.T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
22. Kennedy, J.; Eberhart, R. Particle swarm optimization (PSO). In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
23. Dorigo, M.; Caro, G.D. Ant colony optimization: A new meta-heuristic. In Proceedings of the 1999 Congress on IEEE Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; Volume 2, pp. 1470–1477.
24. Lourenço, H.R.; Martin, O.C.; Stützle, T. Iterated local search: Framework and applications. In *Handbook of Metaheuristics*; Springer: Boston, MA, USA, 2019; pp. 129–168.
25. Zhu, G.; Luo, X.; Miao, Y. Exact weight perfect matching of bipartite graph is NP-complete. In Proceedings of the World Congress on Engineering, London, UK, 2–4 July 2008; Volume 2, pp. 1–7.
26. Mitchell, M.; Holland, J.H.; Forrest, S. When will a genetic algorithm outperform hill climbing. In *Advances in Neural Information Processing Systems*; Morgan Kaufmann: San Mateo, CA, USA, 1994; pp. 51–58.
27. Duhr, S.; Braun, D. Thermophoretic depletion follows Boltzmann distribution. *Phys. Rev. Lett.* **2006**, *96*, 168301. [CrossRef] [PubMed]
28. Creveling, C.M. *Tolerance Design: A Handbook for Developing Optimal Specifications*; Prentice Hall: Upper Saddle River, NJ, USA, 1997.