Path Planning of Automated Optical Inspection Machines for PCB Assembly Systems

Tae-Hyoung Park, Hwa-Jung Kim, and Nam Kim

Abstract: We propose a path planning method to improve the productivity of AOI (automated optical inspection) machines in PCB (printed circuit board) assembly lines. The path-planning problem is the optimization problem of finding inspection clusters and the visiting sequence of cameras to minimize the overall working time. A unified method is newly proposed to determine the inspection clusters and visiting sequence simultaneously. We apply a hybrid genetic algorithm to solve the highly complicated optimization problem. Comparative simulation results are presented to verify the usefulness of the proposed method.

Keywords: AOI machine, genetic algorithm, path planning, optimization, PCB assembly.

1. INTRODUCTION

One important process in electronic manufacturing today is PCB (printed circuit board) assembly using SMT (surface mount technology). SMT has replaced the older through-hole-technology because it can dramatically increase the densities of component per board. Fig. 1 shows a SMT in-line system for PCB assembly. The AOI (automated optical inspection) machine is employed in the SMT in-line systems to perform a sequence of optical inspections on component placement and soldering. Due to the growth of image processing technology, they have become very popular in PCB assembly lines. Efficient operation of the AOI machine is essential for reducing product cost and therefore increasing competitiveness. In this paper, we propose a method to reduce the overall working time of the AOI machine.

The path-planning problem of the AOI machine is to find the optimal path of a camera such that the overall working time is minimized. Since the image acquisition area of the camera is limited by its FOV (field-of-view), all components and soldering pads in the PCB should be divided into many clusters. The size of each cluster should be within the size of the FOV. The camera visits every cluster and acquires an image to perform the PCB inspection. The number of clusters and the moving path of camera have great influence on the total working time. Hence the path-planning problem is to determine the clusters and visiting sequence minimizing the total working time. This problem is under the category of the NP-hard optimization problem, so it is very difficult to find the optimal solution in a reasonable time. In this paper, we propose a method to find the near-optimal solution of the path-planning problem.

Most researches on AOI machines have been focused on image processing [1-3]. It is very difficult to directly apply the typical clustering algorithms [4,5] to our problem because our problem is different from the typical clustering problem in the viewpoint of decision variables and optimization criteria. The optimization method for the TSP (traveling salesman problem) [6] can be partly applied to our problem, but it is necessary to develop a new method to solve the whole path-planning problem.

We can solve the path-planning problem by dividing the whole problem into two sub-problems: clustering problem and sequencing problem. The clustering algorithms are applied to solve the upper stage problem, and then the TSP algorithms are applied to solve the lower stage problem. This hierarchical approach can generate a good solution in a short time, but may generate an inefficient solution since the clustering and sequencing are not independent.

To improve the efficiency of the solution, we newly propose the unified method that solves the clustering problem and sequencing problem simultaneously. The hybrid genetic algorithm is applied to solve the optimization problem. We verify the efficiency and usefulness of the proposed method through the

Manuscript received February 23, 2005; revised August 21, 2005; accepted December 6, 2005. Recommended by Editorial Board member Sangdeok Park under the direction of Editor Keum-Shik Hong. This work was supported by the Regional Research Centers Program of the Ministry of Education & Human Resource Development in Korea.

Tae-Hyoung Park and Nam Kim are with the School of Electrical and Computer Engineering, Chungbuk National University, Cheongju, Chungbuk 361-763, Korea (e-mails: {taehpark, namkim}@chungbuk.ac.kr).

Hwa-Jung Kim is with the R&D Center, KEC Mechatronics Co., LTD, Kumi, Kyungbook 730-030, Korea (e-mail: jjackji@korea.com).

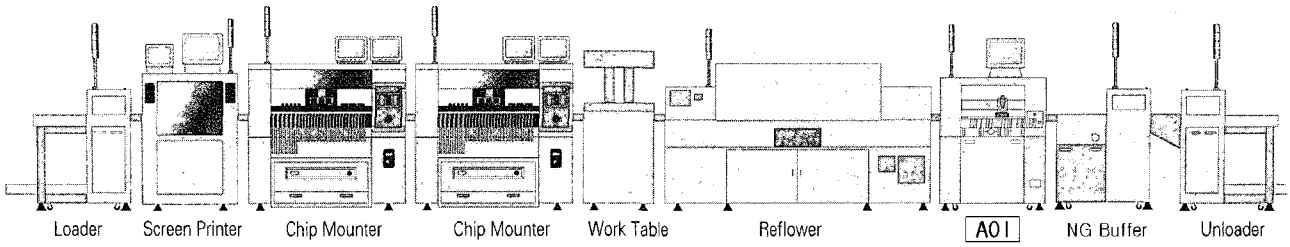


Fig. 1. PCB assembly system and AOI machine.

comparative simulation using a commercial machine.

2. PATH PLANNING PROBLEM

Fig. 2 shows a typical AOI machine that consists of a gantry and a camera. The gantry moves in the y-direction, and the camera moves along the gantry in the x-direction. These x and y-direction movements can occur concurrently. Since the FOV (field-of-view) of the camera is bounded by its limit, the camera travels over the entire board area to acquire overall images.

Fig. 3 depicts inspection windows, FOV and camera path of the AOI machine. The *inspection window* is a rectangular area to be inspected by the camera, which includes component and soldering pad. Several hundreds or thousands of windows are usually located in one PCB. The FOV is the maximum image area that can be acquired by one shot of the camera. The size of the FOV is a constant parameter of the camera, which is usually about several tens of millimeters. The *inspection cluster* is a group of inspection windows that can be captured by one shot of the camera. So the size of the FOV limits that of the inspection cluster. The camera starts from a given initial position, and visits every cluster to acquire image data for all inspection windows. The camera path is the sequence of clusters visited by the camera.

The number of inspection clusters is equal to the number of shots by the camera. Hence if we reduce the number of clusters, we can reduce the total image acquisition time of the AOI machine. The overall working time also includes the camera moving time between clusters. The total moving time is decided by the visiting sequence of the camera. Therefore, inspection clusters and visiting sequence should be determined to reduce the overall working time. The path-planning problem of the AOI machine is to determine the inspection clusters and visiting sequence of the camera.

Now we formulate the path-planning problem mathematically. Let W be a set of window indexes and C be a set of cluster indexes as:

$$W = \{1, \cdots, m\}, \quad (1)$$

$$C = \{1, \cdots, n\}, \quad (2)$$

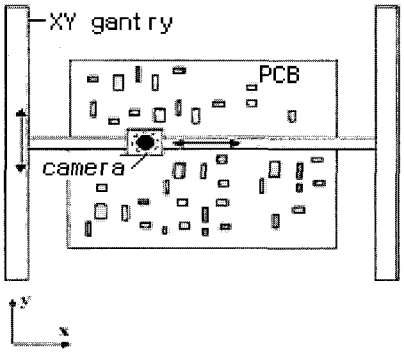


Fig. 2. Structure of AOI machine.

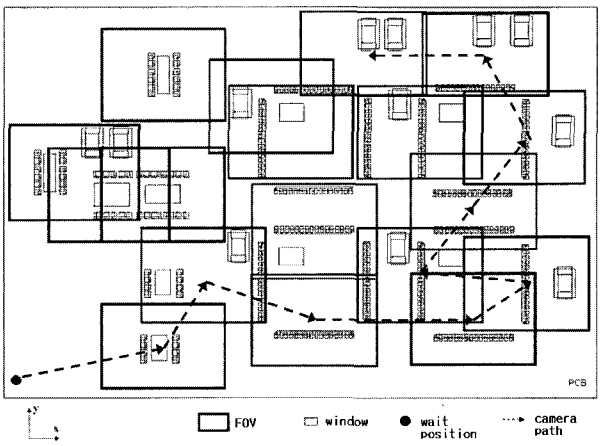


Fig. 3. Inspection windows, FOV and camera path.

where n is the number of clusters, which is a variable to be determined. Define the cluster variable $z_{wc} \in \{0,1\}^{m \times n}$ as:

$$z_{wc} = \begin{cases} 1, & \text{if window } w \text{ is a member of cluster } c. \\ 0, & \text{otherwise} \end{cases}$$

and define the sequence variable $x_{ij} \in \{0,1\}^{n \times n}$ as:

$$x_{ij} = \begin{cases} 1, & \text{if cluster } j \text{ is visited directly from cluster } i. \\ 0, & \text{otherwise.} \end{cases}$$

Then the path-planning problem is formulated as the following integer- programming problem:

$$\min \{n \cdot T_{acq} + \sum_{i \in C} \sum_{j \in C} t_{ij} \cdot x_{ij}\} \quad (3)$$

s.t.

$$\sum_{c \in C} z_{wc} = 1, \ \forall w \in W, \quad (4)$$

$$\sum_{w \in W} z_{wc} \geq 1, \ \forall c \in C, \quad (5)$$

$$X^c_{\max} - X^c_{\min} \leq X_{FOV}, \ \forall c \in C, \quad (6)$$

$$Y^c_{\max} - Y^c_{\min} \leq Y_{FOV}, \ \forall c \in C, \quad (7)$$

$$\sum_{i \in C} x_{ij} = 1, \ \forall j \in C, \quad (8)$$

$$\sum_{j \in C} x_{ij} = 1, \ \forall i \in C, \quad (9)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \ \forall S \subset C. \quad (10)$$

In (3), n is the number of clusters related with the cluster variable z_{wc}. T_{acq} is the image acquisition time for one cluster, which is assumed to be a constant value. And t_{ij} is the camera moving time between cluster i and cluster j, which depends on the cluster variable z_{wc}. Hence the objective function in (3) is the overall working time, which is the sum of total image acquisition time and total moving time.

Constraint (4) means that every inspection window should be included in an inspection cluster, and constraint (5) means that every inspection cluster should include at least one inspection window. $X^c_{\max}$ and $X^c_{\min}$ denote the maximum and the minimum x-coordinates of cluster c, respectively. Also $Y^c_{\max}$ and $Y^c_{\min}$ denote the maximum and the minimum y-coordinates of cluster c, respectively. Therefore constraints (6)-(7) mean that the size of cluster should be smaller than the size of the FOV. Constraints (8)-(9) mean that every cluster should be visited exactly once, and constraint (10) prohibits split cycles from the camera path. Hence the camera path should be the Hamiltonian tour for the traveling salesman problem.

The path-planning problem is to find the cluster variable and the sequence variable such that the objective function (3) is minimized subject to the constraints in (4)-(10). The formulated problem is a nonlinear integer-programming problem with coupled variables. It is known to be very hard to obtain the global solution for the category of these problems. Hence we approach the problem by the local or heuristic method to obtain the near-optimal solution in a reasonable time.

3. HIERARCHICAL METHOD

To overcome the difficulties of the path-planning problem, we divide the overall problem into two sub-problems hierarchically: clustering problem and sequencing problem. The clustering problem is to create the minimum number of inspection clusters, and the sequencing problem is to find the visiting sequence minimizing the total moving time.

The sequencing problem can be modeled as a standard traveling salesman problem (TSP). Therefore we can directly apply the well-known TSP algorithms [6,7] to the sequencing problem. The start node and end node are pre-determined at the wait location of a camera. The cost is the moving time, which can be calculated by profiles of the x and y gantries.

The typical clustering problem [4] is to create clusters to minimize the sum of distances between windows and the center of clusters. The problem does not limit the maximum size of clusters. Also, the number of clusters is usually fixed as a given condition. However, in our clustering problem, the maximum size of clusters should be bounded by the size of the FOV. Furthermore, the number of clusters is not given a condition but a variable to be minimized Therefore it is necessary to modify the typical clustering algorithms for our clustering problem.

3.1. Single-link clustering

The single-link algorithm [4,8] is one of the typical clustering algorithms. It initiates from numerous initial clusters, and merges them together iteratively until the number of clusters reaches a fixed value. This algorithm is very simple to be implemented, and requires low computational complexity. However, the performance of the solution highly depends on the distribution of windows because of local improvement To apply the single-link algorithm to our problem, the size of the FOV should be considered. The modified single-link algorithm is as follows:

S1. Generate initial clusters by setting each window as a cluster.
S2. For each cluster, find the nearest cluster that can be merged together. If the size of the new cluster is within the FOV, merge the two clusters.
S3. Repeat S2 until there is no more merging.

3.2. ISODATA clustering

The ISODATA (iterative self-organizing data analysis) algorithm [4] is the most popular algorithm for the typical clustering problem. This algorithm was updated from the K-means algorithm [9], which improves the clustering by moving the center (mean) of k-clusters. The ISODATA algorithm improves them more efficiently through iterative merging or splitting. Basically this algorithm is also under the category of local improvement method, so the solution depends on the initial clusters. The modified ISODATA algorithm for our clustering problem is as follows:

S1. Generate initial clusters by dividing the board into rectangular grids. The size of each grid is

identical with the FOV.
S2. Delete the cluster in which there is no window.
S3. For each cluster, move the center to include more windows.
S4. If some windows are not included in any clusters, add new clusters for those windows.
S5. For each cluster, find the nearest cluster that can be merged together. If the size of the new cluster is within the FOV, merge the two clusters.
S6. Repeat S3-S5 until there is no more change on clusters.

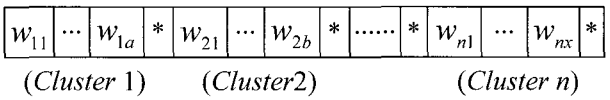
4. UNIFIED METHOD

The hierarchical methods find the cluster variables and the sequence variables at two different stages. However, it is desirable to find the solution simultaneously because both variables are coupled to each other. The proposed method is for the purpose of finding the cluster variables and the sequence variables simultaneously.

The genetic algorithms have been widely used for complex optimization problems. These algorithms can allow the solution to get out of the local problems and approach the global problem [10]. However, convergence of the solution may take a lot of time, which depends on the problem size and parameters. Several researches have been announced for the typical clustering problems [11,12] and TSP [13,14].

Fig. 4 shows the flow of the hybrid genetic algorithm proposed to solve our path-planning problem. To apply the genetic algorithm into our problem, we have to newly define the chromosome, fitness function, and operators.

We define the chromosome of the clustering problem as:



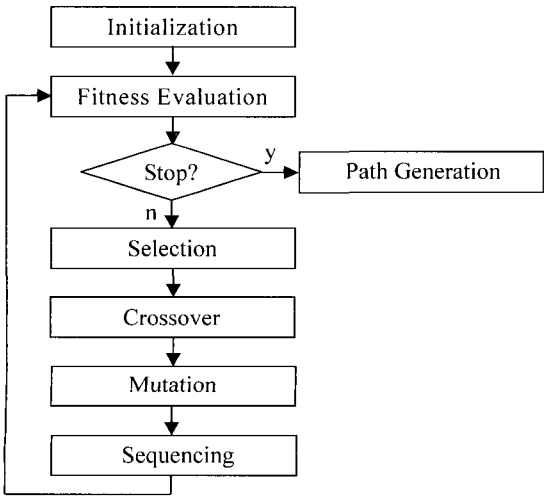


Fig. 4. Flow of hybrid genetic algorithm.

The gene, the element of the chromosome, has the value of window index $w_{ij} \in W$ or mark '*', where w_{ij} denotes the j-th window of the i-th cluster and '*' denotes the end of the cluster. The order of clusters in the chromosome is equivalent to the visiting sequence of the camera. Therefore, one chromosome can represent both the clustering and sequencing results.

4.1. Initialization

Each generation consists of N chromosomes, where N is the population size. The initial population is generated by random number generation.

For each gene of a chromosome, window index $r \in W$ is selected at random. If the size of the cluster including the selected window is within the size of the FOV, the window index is set to the gene. Otherwise, another window index is selected randomly until the feasible condition is satisfied. If a feasible window index is not found, then set mark '*' to the gene. This initialization step starts from the first bit and moves toward the last bit of the chromosome.

4.2. Fitness evaluation and selection

Let t_k be the working time for the k-th chromosome V_k $(k=1,\cdots,N)$, and $t_{\max}$ be the maximum value for all chromosomes in a generation. We define the fitness function f_k for the k-th chromosome as:

$$f_k = (t_{\max} - t_k) \times \frac{N}{\sum_{i=1}^{N} (t_{\max} - t_i)}. \tag{11}$$

The fitness will be 0 for the case of a maximum-time chromosome, and 1 for the case of an average-time chromosome. The value increases as the working time decreases.

The above fitness function is used at the selection stage to perform reproduction of good chromosomes. We adopt the remainder stochastic sampling method [14] to prevent the stochastic sampling error.

4.3. Crossover operator

At the crossover stage, two chromosomes (V_1, V_2) are selected randomly with crossover probability. Then, the crossover operator changes the selected chromosomes to new chromosomes (V_1', V_2') by combining the genes of each chromosome as:

S1. Select a cluster from V_1 randomly.
S2. Search V_2 until all windows of the selected cluster of V_1 are discovered. Then, select the clusters of V_2 if they include any of the discovered windows.
S3. Search V_1 until all windows of the selected cluster of V_2 are discovered. Then, select the clusters of V_1 if they include any of the discovered windows.
S4. Repeat S2-S3 until the selected windows of both

chromosomes are exactly the same.

S5. Exchange the selected clusters between V_1 and V_2, and make new chromosomes V_1' and V_2'.

For example, assume that two chromosomes are selected as:

V_1 = 1 3 * 4 5 10 11 15 * 13 * 16 18 7 8 2 14 * 6 9 12 17 *

V_2 = 1 3 4 6 10 * 18 7 * 8 2 14 * 9 * 13 16 * 5 11 12 15 17 *

In S1, we randomly select an initial cluster from V_1 as:

V_1 = 1 3 * 4 5 10 11 15 * 13 * **16 18 7 8 2 14** * 6 9 12 17 *

In S2, clusters are selected from V_2 as:

V_2 = 1 3 4 6 10 * 18 7 0 8 2 14 * 9 * 13 16 * 5 11 12 15 17 *

In S3, clusters are selected from V_1 again as:

V_1 = 1 3 * 4 5 10 11 15 * 13 * 16 18 7 8 2 14 * 6 9 12 17 *

Since the selected windows of both chromosomes are exactly the same, we go to S5. In S5, the new chromosomes are generated as:

V_1' = 1 3 * 4 5 10 11 15 * 18 7 * 8 2 14 * 13 16 * 6 9 12 17 *

V_2' = 1 3 4 6 10 * 13 * 16 18 7 8 2 14 * 9 * 5 11 12 15 17 *

4.4. Mutation operator

In the mutation stage, one chromosome V_1 is selected randomly with mutation probability. Then, the mutation operator changes the selected chromosome to new chromosome V_1' by random alteration of genes.

The proposed mutation operator is as follows:

S1. Select a cluster from V_1 randomly.

S2. Check whether the windows of the selected cluster can be moved to other clusters. If possible, move them to the feasible clusters and make a new chromosome V_1'.

In S2, the size of the cluster is checked before moving the window since the size of the cluster should be bounded by the FOV. For example, assume that a chromosome is selected as:

V_1 = 3 * 4 17 11 15 * 13 9 6 2 * 16 8 7 1 5 14 * 10 12 *

In S1, we randomly select a cluster from V_1 as:

V_1 = 3 * 4 17 11 15 * **13 9 6 2** * 16 8 7 1 5 14 * 10 12 *

In S2, all selected windows 13, 9, 6, 2 are checked to move. If only two windows 6, 2 can be moved, we move them to the feasible clusters as:

V_1' = 3 **6** * 4 17 11 15 * 13 **9** * 16 8 7 1 5 14 * 10 12 **2** *

4.5. Sequence operator

The sequence operator generates new cluster sequences of the chromosomes whose genes are changed by crossover operator or mutation operator. This operator is also an unary operator for the chromosome.

In the sequence operator, a sequence array is used as auxiliary data. The element of the sequence array is the cluster, and the index of the sequence array denotes the visiting sequence of the cluster.

The proposed sequence operator is as follows:

S1. Set the first cluster of V_1 as a current cluster, and add it to a sequence array.

S2. Find the nearest cluster from the current cluster, where the cluster should not be an element of the sequence queue. Set the nearest cluster as a current cluster, and add it to the sequence array.

S3. Repeat S2-S3 until all clusters are included in the sequence array.

S4. Select a pair of clusters from the sequence array, and exchange the sequence if the total moving time is reduced.

S5. Repeat S4 until all pairs of clusters are selected from the sequence array.

S6. Make a new chromosome V_1' by reassignment of clusters according to the order of the sequence array.

Assume that crossover operator or mutation operator has changed the following chromosome.

V_1 = 3 6 * 4 17 11 15 * 13 9 * 16 8 7 1 5 14 * 10 12 2 *

(*Cluster1*) (*Cluster2*) (*Cluster3*) (*Cluster4*) (*Cluster5*)

The sequence array is generated by S1-S3 as:

(*Cluster 1*) (*Cluster 4*) (*Cluster 2*) (*Cluster 5*) (*Cluster 3*)

And the sequence array is modified by S4-S5 as:

(*Cluster 5*) (*Cluster 3*) (*Cluster 4*) (*Cluster 1*) (*Cluster 2*)

By S6, a new chromosome is generated from the sequence array as:

V_1' = 10 12 2 * 13 9 * 16 8 7 1 5 14 * 3 6 * 4 7 11 15 *

While the crossover operator and mutation operator change both clusters and sequence, the sequence operator changes sequence only. An initial sequence is generated by the nearest neighbour search at S1-S3, and the sequence is improved by 2-opt heuristics [7] at S4-S5. This method is one of the typical local search methods for TSP. The other TSP methods such as 3-opt heuristics or Lin-Kernighan heuristic can also be applied to the sequence operator, but which may result in the lower convergence speed.

The crossover operator and mutation operator guarantee the diversity of search, but the sequence operator helps the fast convergence by local improvement. The hybrid genetic algorithm is the

genetic algorithm that adopts the local search to overcome the problem of convergence speed.

5. SIMULATION

We used a commercial AOI machine (AI-400, Samsung Techwin Co. LTD) [15] for simulation. The size of the FOV was 16(mm)×12(mm), and the image acquisition time for one FOV was 0.25 (sec). The maximum speed and acceleration of the X and Y gantries were 700 (mm/sec) and 0.2 (mm/sec^2), respectively. All PCBs used in the simulation were commercial boards with different number of windows. Table 1 shows the number of windows and size of test PCBs used in our simulation.

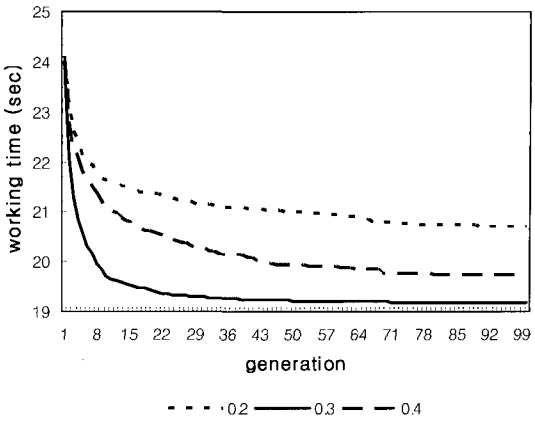
We implemented the proposed algorithms using C++ programming language under MS-Windows XP, and installed them to the off-line programming software of the commercial machine. The population size for the genetic algorithm was set at 100. The crossover probability and mutation probability were set at 0.3 and 0.2, respectively. These parameters were determined by experimental case study. As increasing the population size, the calculation time was increased but the optimization performance was improved.

Fig. 5 presents convergence graphs of the proposed genetic algorithm. According to variations of crossover probability and mutation probability, the best working time in each generation was changed as in Fig. 5(a) and Fig. 5(b). The parameters of the genetic algorithm were selected such that the best working time converged to its minimum value in a short generation.

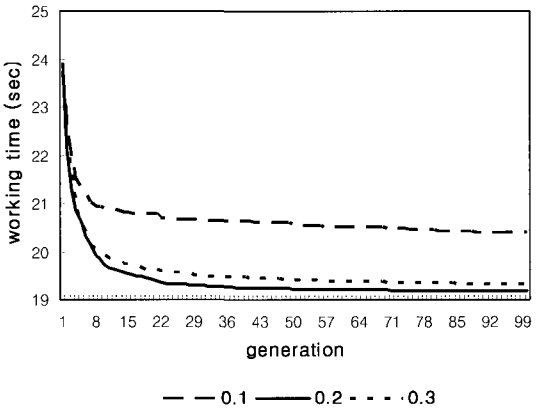
Fig. 6 indicates the results of path planning obtained from different methods. Fig. 6(a) shows a test PCB with 144 inspection widows. Fig. 6(b)-(d) show inspection clusters and camera paths generated by different path-planning methods. Fig. 6(b) was obtained from the hierarchical method with single-link clustering (method 1). Fig. 6(c) was obtained from the hierarchical method with the ISODATA clustering (method 2). Fig. 6(d) was obtained from the unified method using the hybrid genetic algorithm (method 3). Method 2 is the commercial path-planning version of

Table 1. Test PCBs.

PCB Id	No. of windows	PCB size X(mm) x Y(mm)
1	1136	240 x 210
2	1350	240 x 210
3	1578	182 x 284
4	1623	182 x 284
5	1881	282 x 205
6	2049	282 x 205
7	2441	256 x 245
8	3029	256 x 245



(a) Crossover probability (P_c).



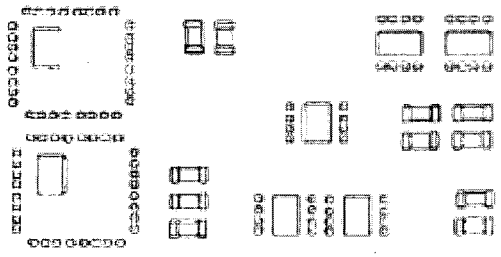
(b) Mutation probability (P_m).

Fig. 5. Convergence graphs for GA parameters.

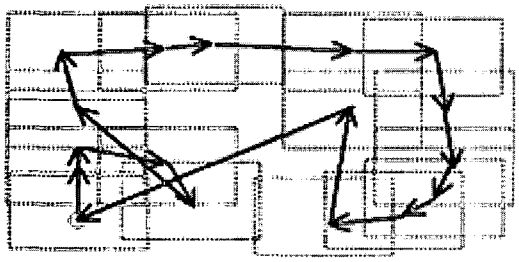
AI-400. Both the number of clusters and distance of the camera path were reduced by method 3, which resulted in the reduction of the overall working time of the AOI machine.

Fig. 7 shows the results of path planning for two PCBs with different distribution of inspection windows. Both PCBs have the same number of inspection windows and the same board size while they have different distributions of inspection windows. Fig. 7(a) and Fig. 7(b) indicate the clusters and sequences generated for PCBs with scattered inspection windows and concentrated inspection windows, respectively. As the inspection windows are scattered, the number of clusters is increased to cover all inspection windows and the overall working time of the AOI is also increased. This simulation shows that the overall working time of the AOI depends on the distribution of inspection windows as well as on the number of inspection windows.

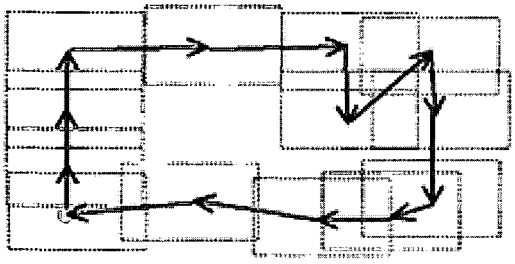
Table 2 presents comparative results of path planning results for the test PCBs in Table 1. The working time is the sum of total image acquisition time and total moving time of the camera. This table shows that the working time strictly depends on the number of clusters, so that clustering is more



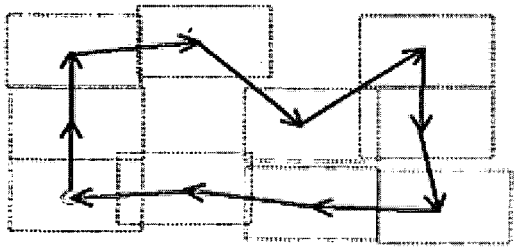
(a) Inspection windows.



(b) Hierarchical method using single-link clustering.



(c) Hierarchical method using ISOSATA clustering.

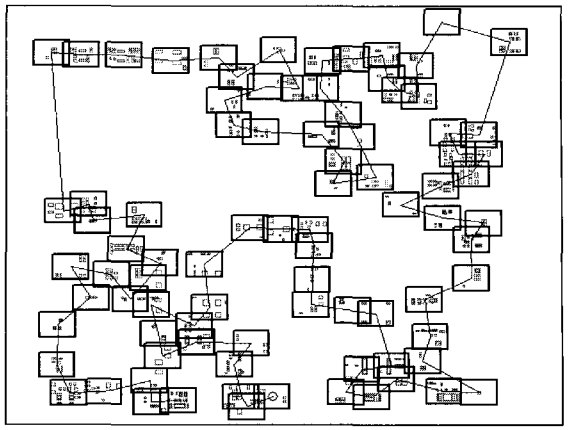


(d) Unified method.

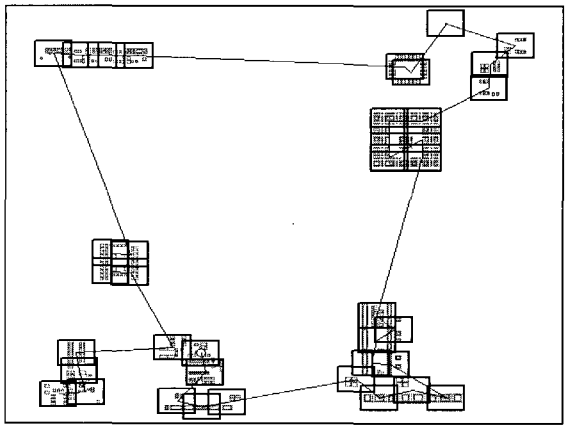
Fig. 6. Examples of path planning by different methods.

important than sequencing in the sense of time optimization. The performance of method 3 is the best among all methods. Table 3 indicates the improvement ratio of method 3 with respect to the other methods. It verifies that the proposed method 3 can improve the performance by about 6 ~ 10%.

Table 4 compares the computational time of each method. The algorithms were run at the Pentium-IV 3GHz computer. The computational time of method 3 is relatively longer than that of other methods. Since the path planning is performed at an offline computer, long calculation time can be allowed if it generates better performance.



(a) PCB with scattered inspection windows. (clusters: 87, working time: 30.1 sec)



(b) PCB with concentrated inspection windows. (clusters: 39, working time: 19.6 sec)

Fig. 7. Example of path planning for PCBs with different window distributions(PCB size: 244(mm) x 216(mm), inspection windows: 1000).

Table 2. Path-planning results: working time.

PCB Id.	Method 1		Method 2		Method 3	
	No. of clusters	Working time (sec)	No. of clusters	Working time (sec)	No. of clusters	Working time (sec)
1	82	30.7	79	29.8	75	27.6
2	85	33.2	81	31.6	77	29.8
3	100	37.0	94	35.2	91	33.5
4	106	38.8	99	37.1	96	34.7
5	114	43.0	110	41.1	104	38.5
6	123	45.3	116	44.2	111	41.1
7	156	57.0	145	54.5	140	50.9
8	173	64.6	165	61.7	163	58.6

Method 1: Hierarchical method using single-link clustering
Method 2: Hierarchical method using ISODATA clustering
Method 3: Unified method

Table 3. Path-planning results: improvement by Method 3.

PCB Id.	W.r.t. Method 1 (%)	W.r.t. Method 2 (%)
1	10.10	7.38
2	10.24	5.70
3	9.46	4.83
4	10.57	6.47
5	10.47	6.33
6	9.27	7.01
7	10.70	6.61
8	9.29	5.02
Ave.	10.01	6.17

Table 4. Path-planning results: computational time.

PCB Id.	Method 1 (sec)	Method 2 (sec)	Method 3 (sec)
1	4.5	6.7	22.2
2	5.3	9.6	33.8
3	6.2	12.9	48.6
4	6.6	14.5	57.0
5	7.5	18.6	79.3
6	9.8	23.8	95.7
7	15.8	33.4	158.8
8	25.6	49.6	220.8

6. CONCLUSIONS

In this paper, we proposed a path-planning method for AOI machines. We defined the path-planning problem and formulated it mathematically. By mathematical formulation, we verified that the problem is a nonlinear integer-programming problem with two coupled variables. To obtain the better solution, we tried to find the two decision variables concurrently. The hybrid genetic algorithm was applied to overcome the problem of local improvement and convergence speed. The chromosome, fitness function and operators were newly defined to solve our problem by genetic algorithm.

The simulation results show that the proposed algorithm can be implemented and installed successfully to practical machines. Also, it can contribute to improve the productivity of the machine by reducing the number of clusters and the camera moving time. The optimization performance of the proposed method was relatively higher than the other methods. The AOI machines have become more popular in the PCB assembly systems, so our results will be useful for increasing the productivity of electronic manufacturing systems.

REFERENCES

[1] T. S. Newman and A. K. Jain, "A survey of automated visual inspection," *Computer Vision and Image Understanding,* vol. 61, no. 2, pp. 231-261, 1995.

[2] Y. Hara, N. Akiyama, and K. Karasaki, "Automatic inspection system for printed circuit boards," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, no. 6, pp. 623-630, 1983.

[3] H. H. Loh and M. S. Lu, "Printed circuit board inspection using image analysis," *IEEE Trans. on Industry Applications*, vol. 35, no. 2, pp. 426-432, 1999.

[4] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264-323, 1999.

[5] B. S. Everitt, S. Landau, and M. Leese, *Cluster Analysis*, Fourth Edition, Arnold, 2001.

[6] G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications*, Springer-Verlag, 1994.

[7] M. Bellmore and G. Nemhauser, "The traveling-salesman problem: A survey," *Operation Research*, vol. 16, pp. 538-558, 1968.

[8] E. Dahlhaus, "Fast parallel algorithm for the single link heuristics of hierarchical clustering," *IEEE Symposium on Parallel and Distributed Processing*, vol. 1, no. 4, pp. 184 -187, 1992.

[9] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 881-892, 2002.

[10] Z. Michalewicz, *Genetic Algorithms+Data Structures=Evolution Programs*, Springer-Verlag, 1996.

[11] L. Y. Tseng and S. B. Yang, "A genetic approach to the automatic clustering problem," *Pattern Recognition*, vol. 34, pp. 415-424, 2001.

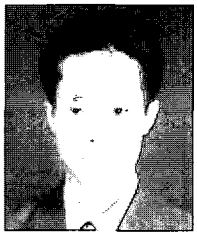
[12] H. S. Kim and S. B. Cho, "An efficient genetic algorithm with less fitness evaluation by clustering," *IEEE Trans. on Evolutionary Computation*, vol. 2, pp. 887-894, 2001.

[13] J. Gu, "Efficient local search with search space smoothing: A case study of the traveling salesman problem," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 24, no. 5, pp. 728-735, 1994.

[14] R. Baraglia, J. I. Hidalgo, and R. Perego, "A hybrid heuristic for the traveling salesman problem," *IEEE Trans. on Evolutionary Computation*, vol. 5, no. 6, pp. 613-622, 2001.

[15] Samsung Techwin Co., LTD, "SMT Inspection equipment - AI400," http://www.samsungtechwin com.

Tae-Hyoung Park received the Ph.D. in Control and Instrumentation Engineering from Seoul National University, Korea in 1994. From 1994 to 1997, he was a Senior Research Engineer at Samsung Techwin Co., where he developed the optimization software for PCB assembly systems. Since 1997 he has been a Professor of Control and Instrumentation Engineering at Chungbuk National University. From 2000 to 2001, he was a Visiting Professor at the University of Toronto, Canada. His research interests include robotics, optimization algorithms, and electronic manufacturing systems.



Hwa-Jung Kim received the M.S. degree in Control and Instrumentation Engineering from Chungbuk National University, Korea in 2004. He is currently a Research Engineer in the Mechatronics Laboratory at KEC Co. His research interests include robotics, optimization algorithms, and machine vision.

Nam Kim received the Ph.D. degree in Electronics Engineering from Yonsei University, Korea in 1988. Since 1989, he has worked as a Professor in the Dept. of Computer and Communication Engineering, Chungbuk National University. From 1992 to 1993, he was a Visiting Professor at Stanford University. From 2000 to 2001 he was a Visiting Professor at Caltech. He is interested in the applications of holography, diffractive optics, optical interconnection, and the optical memory system.