Proceedings of the
2000 Japan•USA Flexible Automation Conference
July 23-26, 2000, Ann Arbor, Michigan

2000JUSFA-13124

OFF-LINE ERROR RECOVERY LOGIC SYNTHESIS IN AUTOMATED ASSEMBLY LINES BY USING GENETIC PROGRAMMING

Cem M. Baydar
(cbaydar@umich.edu)

Kazuhiro Saitou
(kazu@umich.edu)

Department of Mechanical Engineering and Applied Mechanics,
The University of Michigan, Ann Arbor, USA

ABSTRACT

Unexpected failures are one of the most important problems, which cause costly shutdowns in an assembly line. Generally the recovery process is done by the experts or automated error recovery logic controllers embedded in the system. The previous work in the literature is focused on the "on-line" recovery of the assembly lines which makes the process, time and money consuming. Therefore a novel approach is necessary which requires less time and hardware effort for the generation of error recovery logic. The proposed approach is based on three-dimensional geometric modeling of the assembly line coupled with the evolutionary computation techniques to generate error recovery logic in an "off-line" manner. The scope of this work is focused on finding an error recovery algorithm from a predefined error case. An automated assembly line is virtually modeled and the validity of the recovery algorithm is evaluated in a "generate and test" fashion by using a commercial software package. The obtained results showed that the developed framework is capable of generating recovery algorithms from an arbitrary part positioning error case. It is aimed that this approach will be coupled with the error generation in the future, providing efficient ways for the study of error recovery in automated assembly lines.

Keywords: Error Recovery Synthesis, Genetic Programming, Off-line Programming, Automated Assembly Lines.

INTRODUCTION

An unexpected failure is an unavoidable phenomenon, which causes the automated assembly lines to halt their operation. These failures can bring out drastic results in economical issues. As indicated in the results of EUREKA (Luxhoj *et al*., 1997) project, initiated to benchmark maintenance in Scandinavian countries in 1992, approximately 30% of the time spent on maintenance is used for unforeseen repairs, 20% for preventive maintenance and 37% for planned repairs. A similar survey in the United States showed that excessive maintenance costs were approximately 200 billion dollars in 1990.

The diagnosis and recovery from such failures are normally handled by *"on-line"* investigation of the assembly line by the experts, which means costly shutdown of the assembly lines. In automated systems (Zhou and DiCesare, 1989), up to 90 % of the control coding effort is based on error recovery by using Programmable Logic Controller (PLC) codes. However these PLC codes are programmed by humans based on *"expected"* error scenarios and they are deficient in dealing with unforeseen, *"unexpected"* scenarios, leaving the recovery process to manual labor work. A novel approach to deal with the unexpected failures is *"off-line"* synthesis of error diagnosis and recovery logic based on the three-dimensional geometry based modeling of an entire assembly line. Generation of *"unexpected"* error cases can be accomplished by using Monte Carlo simulation of the assembly process based on the statistical model of the dimensional and functional errors in sensors, robots and products. Once those *"unexpected"* cases are generated, error recovery logic synthesis for those cases can be studied. As an initial step, this paper is focused on the second part of the proposed approach, mainly automated synthesis of error recovery logic from a predefined error case.

It is stated in (Visinsky *et al*., 1994) that in an assembly line, most errors occur during the part transport and part mating. However since the part presentation errors are mostly dependent on the nature of the assembly line (Lopes and Camarinha-Matos, 1994), there may be different ways of recovery. The objective of this study is to obtain an error

recovery algorithm for a part presentation error in a given three-dimensionally modeled assembly line. The solution of this problem consists of a special computer program, which is composed of several commands and directions for the industrial robot. This program can be downloaded to the robot controller to perform the recovery task. These commands in the recovery program are chosen from available set of commands which makes the problem a discrete decision making process. The generation of the optimal recovery program is done by a heuristic-search (Koza, 1992) among the alternative error recovery programs and it is called Genetic Programming. Genetic Programming (GP) (Banzhaf *et al*., 1998) aspires to do precisely that – to induce a population of computer programs that improve automatically as they experience the data on which they are trained.

The method of finding the optimal computer program is done in a "*generate and test*" fashion and the validity of the generated programs is tested with a commercial software package (Workspace, 1998) called Workspace from Flow Software Inc.

PREVIOUS WORK

Srinivas (Srinivas, 1977) is one of the earliest researchers who investigated error detection and recovery strategies. His approach was considering the tasks to be decomposable into a sequence of transformations from the initial state to a goal state. Each of the states between the initial state and the goal state is monitored. If a state has not been reached, this means a failure is occurred. The next step is building a failure tree and generating an error recovery plan.

Expert systems are one of the most popular tools used in the error detection and recovery in flexible assembly systems. Two different methods are used in the literature. The first method (Brnyjolfsson and Arnstrom, 1990) uses an expert system to monitor robot operations and if detects an error, the robot stops. The expert system goes into an error diagnostic mode and analyses the sensors that describe the environment. After finding the cause of the error, it recommends a solution procedure. The second method (Abu-Hamdan and El-Gizawy, 1997) uses the expert system to plan and execute the assembly process. The user enters a part oriented description of the assembly. The expert system generates an assembly plan and if an error occurs, it revises the plan.

Tzafesas and Stamou's approach (Tzafestas and Stamou, 1997) is based on using knowledge-based approaches for automated assembly. It is believed that fuzzy logic and fuzzy reasoning-based techniques are more adequate than the others under specific circumstances. The assembly system is trained on-line for the possible errors and appropriate recovery logic rules are embedded to the system. Recovery is accomplished by making deductions out of these rules. Telagi and Soni (Telagi and Soni, 1994) investigated the different control methodologies such as neural networks and fuzzy logic in manufacturing environment. Evans and Lee (Evans and Lee, 1994) developed a method to integrate reactive planning for automated error recovery while Kao (Kao, 1995) discussed the selection of an optimal error recovery strategy among the given recovery options using a semi-Markovian model of production states during recovery operations.

Several work have also been done on how to manipulate the PLC code safely with the error recovery codes generated manually or automatically without introducing new errors. Zhou and DiCesare (Zhou and DiCesare, 1989) proposed four argumentation methods of process control logic code with error recovery codes: input conditioning, alternate path, feedback error recovery and feedforward error recovery. Cao and Sanderson (Cao and Sanderson, 1992) proposed a fuzzy Petri-net controller as an integrated representation of process control logic and error recovery. However, those approaches are lack of handling geometric features of the assembly line, which is essential to make predictions of error scenarios.

All of the methods discussed above are focused on the *"on-line"* recovery and training of the assembly lines which makes the process, time and money consuming. Therefore a novel *"off-line"* approach is necessary which requires less time and hardware effort for the generation of the error recovery logic.

PROPOSED METHOD

For robots to execute the programs generated off-line successfully, the dimensions of the real-world components need to be modeled accurately. Otherwise, the difference will lead to positioning and orientation errors. The method presented here is based on accurate three-dimensional geometric model of the assembly line by using a commercial software package. This three dimensional model provides the framework of testing the generated error recovery logic "off-line" in less time than the conventional *"on-line"* methods.

In this study, during the generation of the error recovery logic for a predefined error state, Genetic Programming is used. The term "Genetic Programming" was first introduced by Koza (Koza, 1992) in 1992 and it addresses the problem of automatic programming namely, the problem of how to enable a computer to do useful things by automatic programming (Banzhaf *et al*., 1998). It uses the working principles of Genetic algorithms (GAs).

Genetic algorithms were first introduced by Holland (Holland, 1975) in 1975. In GAs, design variables are coded onto fixed-length or variable-length strings that are analogous to chromosomes in biological systems (Goldberg, 1989). Strings are composed of characters, which are analogous to genes. Each string represents a solution point in the search space. An objective function is defined within the problem and the GA tries to maximize the fitness of a solution point based on a fitness function related with the objective. In GA, two basic operators are applied to the selected pairs. First operator is called "crossover" in which the strings of two members are cut and recombined from a random point, producing two new members. The second operator is called "mutation" and it is

applied by selecting a place in the string randomly and changing its value. Mutation has the advantage of introducing some diversity into the search while crossover uses the properties of the current population to combine and produce better results.

In Genetic Programming, each member in the population is a computer program for the solution of the problem. The reason of selecting GP as an optimization method in this work is its power of dealing with discrete design optimization problems as well as the ease of its implementation in this case. Since an optimal computer program is being searched, GP is a perfect choice to be used in such a problem.

A software module, which is responsible from the evolutionary computation process, is developed and implemented to the system. This program enables the user to generate programs for the problem and after that evaluate the outputs form the commercial software and proceed with the evolution process for the next generations. The basic working mechanism is given in Figure 1.

Error recovery algorithm is basically a computer program, which is responsible from the recovery process and composed of several specific commands. These commands (Workspace, 1998) move the robot arm to the specified point (taking the reference as the robot base and the destination point as the tool center point) or move the arm relative to a predefined coordinate point. The commands used in this study are from KAREL2 Robot Language and they are Move To, Move Away, Move Near and Move Relative.

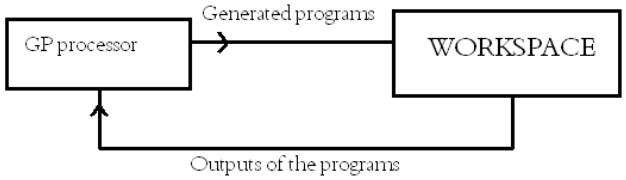

Figure.1: Framework of the Optimizer

The use of program commands requires a formal way of combining with the variables for Tool Center Point (TCP), relative movement in the negative z-direction (Offset) and relative movement in 3D space (Vector). The following table shows the appropriate lexical representation for each command.

Table 1: The use of Commands

Command	Suffix
Move To	<TCP>
Move Away	<Offset>
Move Near	< TCP> By <Offset>
Move Relative	<Vector>

TCP variable is composed of 6 sub-variables. These variables are the three coordinate points in the robot frame and three orientations of the tool holder arm. A vector is composed of three offset variables in the space. This variable is coupled with the Move Relative command enabling the robot to move its arm relatively from its current position.

The objective is to minimize the part placement error between the final position and its optimal position on the fixture. Therefore if a distance function is used between the recovered position and the optimal position, the problem can be stated as a single objective optimization problem such as:

$$Minimize \ \sqrt{(x-x_o)^2+(y-y_o)^2+(z-z_o)^2} \tag{1}$$

The global optimum is located at (x_o, y_o, z_o). However, in this study the modeled system allows the workpiece to be placed in tolerance limits of the optimum place so a local optimum is also acceptable. Therefore, in all dimensions the acceptable tolerances are taken as 5 mm. This enables to simplification of the problem against the computational cost since it is easy to converge a local optimum. Therefore the following constraints are added to the problem:

$$|x-x_o| \leq 5 \tag{2}$$
$$|y-y_o| \leq 5 \tag{3}$$
$$|z-z_o| \leq 5 \tag{4}$$

All of the physical constraints (collisions between the objects, restrictions due to robot kinematics, etc.) are defined within the software package and evaluated with its internal engine. After the problem is defined as an optimization problem, the infrastructure for the evolutionary computation side is developed.

A recovery program is composed of lines and each line is composed of commands and appropriate suffixes. A chromosome structure is defined for each line. Each program is composed of several strands of chromosome and maximum is limited to 10. For each chromosome the values of the variables are taken from a set, which is randomly generated in the beginning. These values are stored in one main matrix and several sub-matrices in order to be retrieved and processed during the evolution process.

The fitness function is defined as the inverse value of the objective function and the problem is turned out to be a maximization problem. In each generation all of the members of the population are evaluated with respect to the desired fitness function. The crossover probability is taken as 0.9. This means that 90% of the population participates in the crossover sequence. The probability of a member to participate in the

crossover is directly proportional with its fitness value and it is selected randomly, which is also called weighted roulette wheel selection. The probability for a member to be in the mate pool is given as:

$$p_i = \frac{f_i}{\sum f_i} \tag{5}$$

f_i is the fitness of the individual. After two parents are selected the crossover operation takes place. The first step in the crossover process is locating the crossover line point between the programs. After the crossover line is determined, the next step is locating the crossover point on the chromosome structure. A point on the chromosome is selected randomly and two children are generated in the next generation by exchanging the genes after the crossover point.

Mutation takes place after the crossover operation in order to prevent premature convergence. By applying mutation, a small portion of new members is introduced to the population. The mutation probability changes between 0.015 and 0.05 depending on the nature of the population.

After the evolution step, the new generation of the programs is stored in the main matrix structure. This encoded structure is decoded into working programs in KAREL2 language. These programs are tested in Workspace and their outputs are written into text files. After that, these outputs are processed with the developed program and next evolutionary step takes place according to these outputs. The maximum number of generations is limited to 50 as it is suggested in (Banzhaf *et al*., 1998) as a rule of thumb, while the population size in one generation is restricted to 100 for initial results. It is observed that this size of population is adequate from the case studies. Figure 2 summarizes the steps of processing.

CASE STUDIES

Several case studies are conducted with the model to test the efficiency and validity of the developed system. An automated assembly line model is constructed with the commercial software package. This line consists of four robots, which are responsible from the transportation and welding operation of two sheet-metal parts. Our pre-defined *"unexpected"* error scenario is the collision of the workpiece with the environment during its transportation from the conveyor to the fixture. An IRB6001 type 6-axis industrial robot is responsible for the presentation of the workpiece to the fixture. Figure 3 shows the ideal positioning of the workpiece with the fixture. The desired orientation of the workpiece is known and this orientation is to be restored at the end of the recovery program.

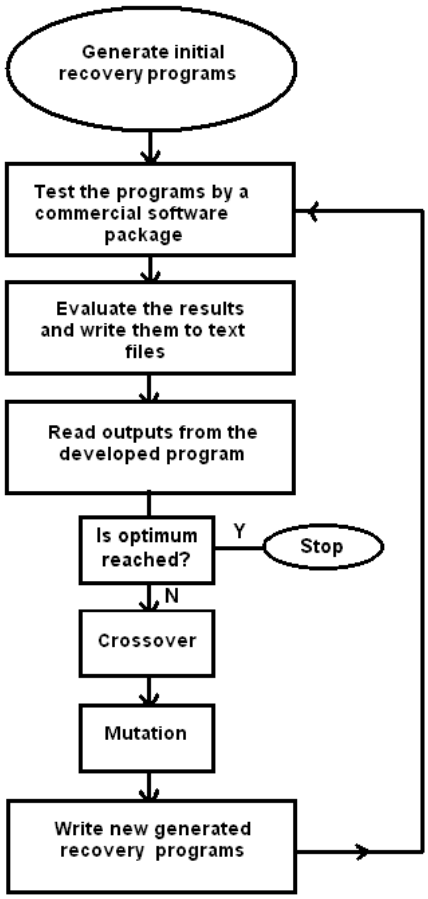

Figure.2: Working Mechanism of the Developed Framework

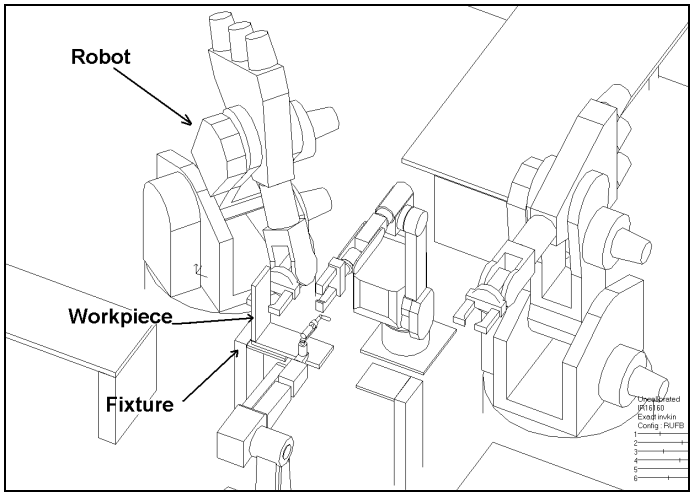

Figure.3: Ideal Positioning of the Workpiece

During the case studies it is assumed that after the collision, the part is still held in the gripper and the sensor on the fixture is working properly in order to detect the correct positioning.

Case Study 1

The first case study is taken from a collision state occurred between the workpiece and the fixture, shown in Figure 4. Several runs are carried with the developed infrastructure. A cubical space for the robot movement is defined as the envelope, which has the size of 400x400x400 mm^3. As it is observed from the outputs (Table 2), an error reduction of 65% is gained in the replacement of the workpiece after the initial generation. In generation 11, an optimal result is found since all the final position of the workpiece is in the limits of 5 mm tolerance placement. This is a local optimum however acceptable since it is in the tolerance limits.

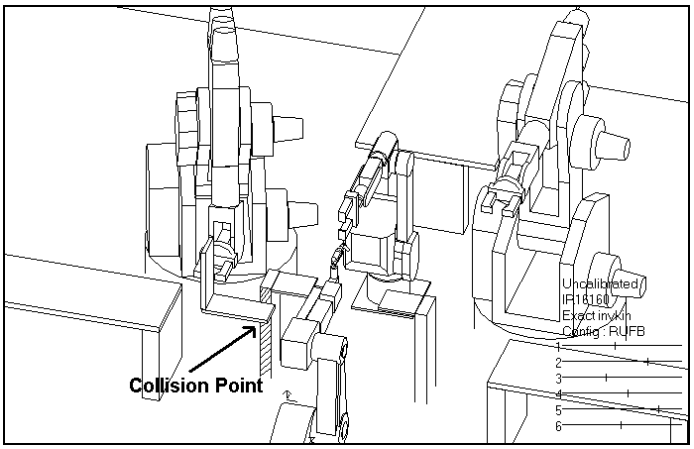

Figure.4: Collision case between the workpiece and the fixture

The best program obtained after the 10^{th} generation is composed of the lines stated below. The last command line is added automatically by the system in order to restore the desired final orientation. The resulting placement errors are given in Table.3.

Table.2: Outputs from the 1st Case Study

Generation	Objective Function
1	24.535
2	31.968
3	14.866
4	14.866
5	14.212
6	13.967
7	12.180
8	12.180

Table.2: Outputs from the 1st Case Study (continued)

9	11.682
10	9.708
11	8.66

‘ Best Program of Case Study 1:
ROUTINE GPCode26
BEGIN
Move To POS (-710, -684, -982, 90, 80, 0,’RUFB’)
Move Relative (9,8,10)
Move To POS (-701 –692, -992, 90, 90, 0,’RUFB’)
END GPCode26

During the recovery process, it is observed that the algorithm is composed of three lines only between ‘BEGIN’ and ‘END” commands. The maximum lines are limited to 10 and this result shows that recovery is accomplished in significant small number of steps.

Table.3: Placement Errors

Coordinate:	Error (mm):
X:	5
Y:	5
Z:	5

The history of the optimization is given in Figure 5. It is observed that both the fitness values of the best and the worst recovery programs are increasing as the evolution takes place. It should be noted that the fitness function (to be maximized) in the inverse of the objective function (to be minimized).

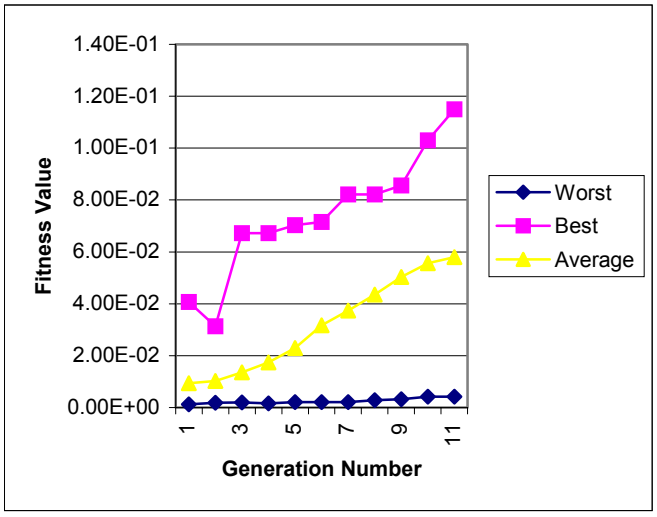

Figure.5: Optimization Progress of the 1st Case Study

Case Study 2

In the second case study, the effect of expanding the working envelope of the robot is experimented. Same collision point is used from the first case and the envelope is expanded to 5 times. It is expected that this approach would get better results since much freedom is left for the robot movement.

It is observed that 91% reduction is obtained relative to the initial generation and the performance is increased relative to the first case study. However in this case it took 14 generations to obtain the optimal result. Table 4 shows the objective function values obtained after each generation.

Table.4: Outputs of the 2nd case study

Generation	Objective Function
1	55.652
2	48.093
3	57.775
4	31.874
5	23.706
6	19.647
7	19.027
8	19.027
9	21.471
10	21.471
11	19.647
12	12.45
13	10.321
14	5.196

The best program is composed of three lines and it has 3 mm error in the replacement after the recovery process as shown in Table 5.

‘Best Program of Case Study2:
ROUTINE GPCode14
BEGIN
Move To POS(-710, -684, -982, 90, 80, 0,'RUFB')
Move To POS(-703, -694, -990, 20, 90, 0,'RUFB')
Move To POS(-703, -694, -990, 90, 90, 0,'RUFB')
END GPCode14

Table.5: Placement Errors

Coordinate:	Error (mm):
X:	3
Y:	3
Z:	3

In this case, a similar way of recovery as in the first case is obtained. At first the robot carries the workpiece to the same point as in Case 1. Then places it to a place closer than before and it is in acceptable tolerance. The progress of optimization is shown in Figure 6.

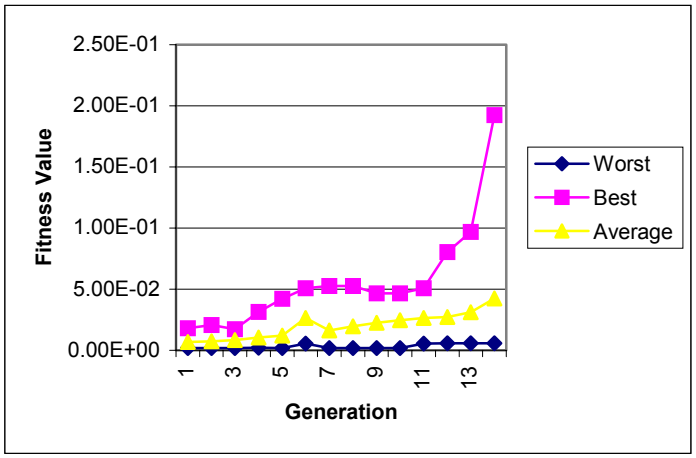

Figure.6: Optimization Progress of the 2nd Case Study

Case Study 3

For the third case study, the collision point is changed in order to test the robustness of the developed system. Figure 7 shows the new collision scenario between the workpiece and the fixture.

Similar to the previous results, this time it took 13 generations to reach the optimum results. A reduction of 87 % is gained relative to the initial generation. Table 5 shows the progress of decreasing in the objective function.

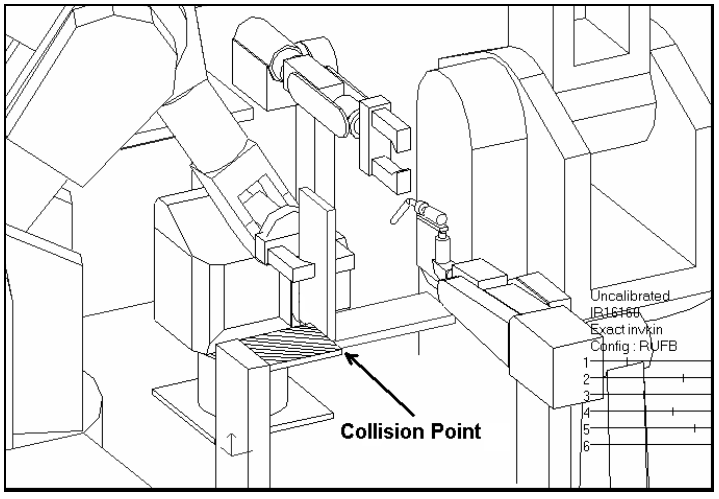

Figure.7: Positioning error in the 3rd case study

Table.5: Outputs of the 3rd case study

Generation	Objective Function
1	50.577
2	35.525
3	33.136
4	17.972
5	17.972
6	12.884
7	12.884
8	12.884
9	12.884
10	12.884
11	12.884
12	9.210
13	6.708

The placement errors are acceptable and given in Table 6. It is observed that similar to the previous cases, the recovery algorithm is composed of three lines. This time robot carries the workpiece to a different initial point when the recovery procedure starts.

Table.6 Placement Errors

Coordinate:	Error (mm):
X:	4
Y:	3
Z:	4

'Best Program of Case Study3:
ROUTINE GPCode26
BEGIN
Move To POS (-695, -670, -987, 90, 30, 0,'RUFB')
Move Away -36
Move To POS (-710, -700, -991, 90, 90, 0,'RUFB')
END GPCode26

The optimization progress is given in Figure 8. Although the fitness value of the best member is same between the 6th and 11th generation, in the end it has converged to an acceptable optimum value. It is observed that the average value increased during these generations, which eventually gave a "jump" to the best fitness value.

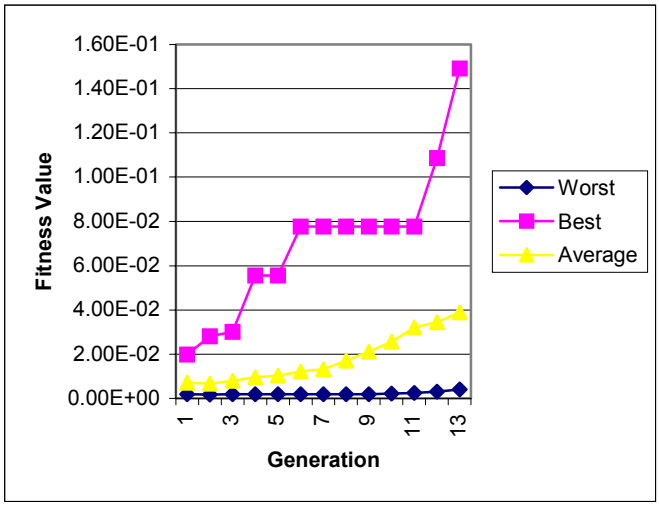

Figure.8: Optimization Progress of the 3rd Case Study

The case studies demonstrated that the developed framework is efficient for generating recovery logic and flexible to handle different positioning errors.

DISCUSSIONS AND FUTURE WORK

Automated assembly lines are subjected to unexpected failures during their normal operation. Such failures may lead to costly shutdowns and generally require *"on-line"* diagnosis and recovery by the experts in order to restore their normal operation. Previous methods for the automated recovery are focused on the "*on-line"* recovery and training of the assembly lines. In this paper a different approach, which uses three dimensionally modeled systems, is proposed. A computer program is developed and coupled with a commercial software package to work *"off-line"* on the error recovery logic generation by using evolutionary computation. Several case studies are performed on a virtually modeled assembly line and they are demonstrated that the developed system is efficient to find the optimum recovery logic from an arbitrary collision case, which is originated from a part placement error.

It is expected that this approach will require less time and labor effort for the generation of the error recovery logic and provide efficient ways for the study of error recovery in automated assembly lines. The possible directions for the future work include determining the robust recovery algorithms from *multiple error cases* and *generation of unexpected error cases* from the assembly model by using Monte Carlo simulation of the assembly process based on the statistical model of the dimensional and functional errors in sensors, robots and products.

ACKNOWLEDGMENTS

The authors are grateful for the technical support of Flow Sofware Inc. for the Workspace™ software. The authors also gratefully acknowledge The University of Michigan scholarship awarded to the first author for his graduate studies at The University of Michigan.

REFERENCES

Abu-Hamdan M. G., El-Gizawy A. S., *"Computer Aided Monitoring System for Flexible Assembly Operations"*, Computers in Industry, Vol. 34, pp. 1-10, 1997.

Banzhaf W., Nordin P., Keller R., Francone F., "Genetic Programming: An Introduction", pp 4-5, Morgan Kauffman Publishers Inc., 1998.

Brnyjolfsson S., Arnstrom A., *"Error Detection and Recovery in Flexible Assembly Systems"*, The International Journal of Advanced Manufacturing Technology, Vol.5, pp. 112-125, 1990.

Cao T.C., Sanderson A.C., *"Sensor-based error recovery for robotic task sequences using fuzzy petri-nets"*, Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Vol.2, pp.1063-1069, 1992.

Evans E.Z., Lee S.G., *"Automatic Generation of Error Recovery Knowledge Through Learned Activity",* Proceedings of the 1994 IEEE International Conference on Robotics and Automation, Vol. 4., pp. 2915-2920, 1994.

Goldberg D., "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, Reading, MA, 1989.

Holland J., "Adaptation in Natural and Artificial Systems", MIT Press, Cambridge, MA, 1975.

Kao J.F., *"Optimal Recovery Strategies for Manufacturing Systems*", European Journal of Operations Research, Vol.80, pp.252-263, 1995.

Koza J. R., "Genetic Programming: On the Programming of Computers by Natural Selection", MIT Press, Cambridge, MA, 1992.

Lopes L.S and Camarinha-Matos L.M, *"Learning to Diagnose Failures of Assembly Tasks"*, Artificial Intelligence in Real Time, pp 97-103, 1994.

Luxhoj J.T., Riis J. O., Thorsteinsson U., *"Trends and Perspectives in Industrial Maintenance Management"*, Journal of Manufacturing Systems, Vol.16, No.6, 1997.

Srinivas S., "Error Recovery in Robot Systems", Ph.D Thesis, California Institute of Technology, 1977

Telagi S. M., Soni A. H., *"Control Systems in Manufacturing"*, Proceedings of 1994 ASME Design Technical Conferences – 4th ASME Flexible Assembly Conference, Vol. 73, pp.17-23, 1994.

Tzafestas S. G., Stamou G. B., *"Concerning Automated Assembly: Knowledge-Based Issues and a Fuzzy System for Assembly under Uncertainty"*, Computer Integrated Manufacturing Systems, Vol. 10, No.3, pp. 183-192, 1997.

Visinsky M.L., Cavallaro J.R., Walker I. D., *"Expert System Framework for Fault Detection and Fault Tolerance in Robotics"*, Computers in Electrical Engineering Vol. 20, No.5, pp 421-435, 1994.

Workspace v.4 Educational User Guide Manual, 1998.

Zhou M. C., DiCesare F., *"Adaptive design of petri-net controllers for error recovery in automated manufacturing systems"*, IEEE Transactions on Systems, Man and Cybernetics, Vol.19, No.5, pp.963-973, 1989.