

# A Genetic Algorithm for Configuring Reconfigurable Conveyor-Components in a Flexible Assembly Line System

*Tobby K W To & John K L Ho*

Department of Manufacturing Engineering & Engineering Management  
City University of Hong Kong

Tat Chee Avenue, Kowloon, Hong Kong SAR, P.R. China  
Fax: (852) 2788 8017 E-mail: [tobby.to@plink.cityu.edu.hk](mailto:tobby.to@plink.cityu.edu.hk)

## ABSTRACT

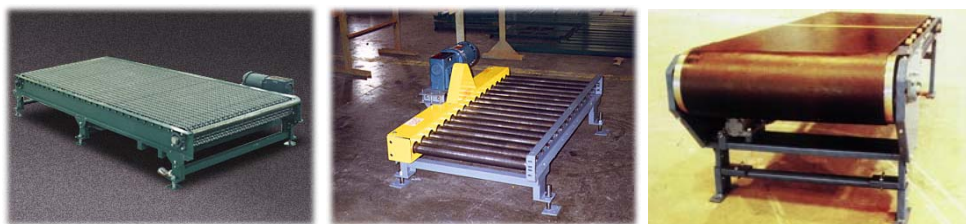
Many reconfigurable conveyor-components have been developed for the construction of assembly line systems. The components have different transporting paths, shapes, sizes, and etc. This is very difficult to establish a good system configuration among all possible assembly line configurations. This paper describes a genetic algorithm (GA) to configure those reconfigurable conveyor-components forming a flexible assembly line system (FALS) to meet the ever-changing production requirements. The transporting paths, shapes and sizes of reconfigurable conveyor-components are coded into binary string as chromosome to represent an assembly line layout for analysis and evaluation. The layouts are generated by the three evolutionary processes: selection, crossover, and mutation. The process of updating control parameters is integrated into the GA to improve the performance and efficiency of the evolutionary processes. The reconfiguration of a FALS to meet the requirements of minimization of the number of reconfigurable conveyor-components and the provision of alternative processes paths are discussed in details in the paper.

## 1. INTRODUCTION

Nowadays, the customer-driven products are very diversified. Facing of this manufacturing situation, many factories have attempted to introduce flexible assembly line systems (FALS) as the strategy to produce the diversified products. Recently, many reconfigurable conveyor-components have been developed. Typical reconfigurable conveyor-components as shown in Figure 1 are linear conveyor, rotating conveyor, conveyor-bend, S-shape conveyor, U-shape conveyor, and lift conveyor. They have different transporting paths, shape, sizes, and etc. Those reconfigurable conveyor-components can be formed into various assembly line configurations. As the result, a number of design alternatives may exit and many possible system configurations can be formed to meet the production needs. This is very difficult to reconfigure an assembly line system among all possible configurations. The technique of genetic algorithm (GA) is introduced in section 3.3 to overcome the difficulty of the reconfiguration of a FALS.

**Figure 1. The Typical reconfigurable conveyor-components**

Type 1: Linear Conveyor



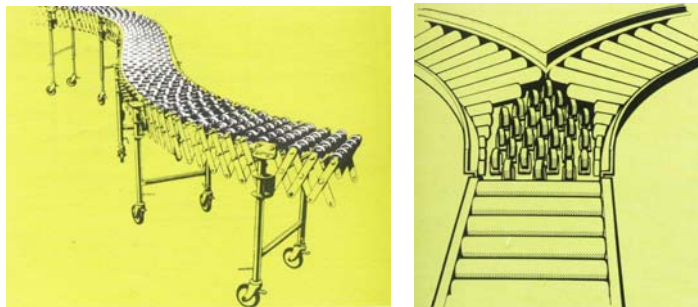
Type 2: Rotating Conveyor



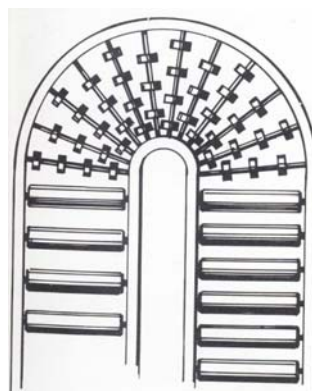
Type 3: Conveyor-bend



Type 4: S-Shape & Y-Shape Conveyors



Type 5: U-Shape Conveyor



## Type 6: Lift Conveyor



## 2. RECONFIGURABLE CONVEYOR-COMPONENTS

Figure 1 shows the typical conveyor-components. They are widely used to transport work-parts in an assembly line system.

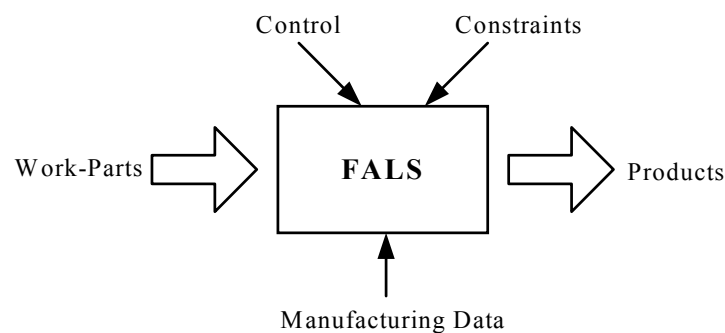
Linear conveyor is used for transporting the work-parts in linear direction. The length and speed of the conveyor usually can be adjusted. Rotating conveyor is mainly used to change the moving direction of work-parts. It can lift and rotate the work-parts to different altitude and different orientation (Ho *et al*, 1997). Conveyor-bend is used to change the moving directions of work-parts. The bend always consists of rollers or wheels, in general, with standard angles of 30°, 45°, 60°, and 90° (Lindkvist, 1985). S-shape, Y-shape, and U-shape conveyors always consist of wheels and rollers, which are designed to flex side ways, to expand and contract for specific routing. Y-shape conveyor is widely used in a mixed-model assembly line system. U-shape conveyor usually appears in a closed loop assembly line system. Lift Conveyor is designed for three-dimensional conveyor system to transport the work-parts at different levels.

## 3. RECONFIGURATION OF FALS

### 3.1. Information required for system reconfiguration

The major function of a FALS is to assemble a number of work-parts into a product. The information required for the reconfiguration of a FALS can be categorized in three aspects: manufacturing data, controls, and constraints as shown in Figure 2.

Figure 2. The required information for the configuration of a FALS



Manufacturing data provide the information that directly concern with the overall assembly process such as:

- The number of assembly workstations
- The location of the assembly workstations, loading and unloading stations
- The processing time of each workstation
- The size of the reconfigurable conveyor-components
- Production volume that determines the throughput of an assembly line
- Production lead-time that determines the total amount of time for assembly

The controls provide the information that directly control the movement of a work-part such as:

- The assembly processing sequence;
- The speed and the transporting direction of the work-part
- The required orientation of the work-part for assembly
- The position of the work-part for assembling operation taking place

Constraints specify the conditions of being limited for the design of an assembly line such as:

- The space restriction
- The restriction of the assembly equipments
- The restriction of the conveyor-components

### **3.2. Recent R&D work**

Currently, a number of methods using knowledge-based system (KBS) technique for reconfiguring assembly line systems have been developed. Myung and Han (2001) integrated knowledge-based system with computer-aided design (CAD) to develop a computer-aided configuration design method for redesigning assemblies of machine tool. Zha, Du, and Lim (2001) developed knowledge Perti net approach for the design of automation assembly system. The approach is to reduce the development time of setting up an assembly system. Leung (2000) developed a knowledge-based system to make FALS to be more intelligent meeting the defined needs and the unforeseeable needs for the given recourse.

KBS technique is based on the collected knowledge and expert's experience to search a possible layout. It may generate poor results due to utilization of incorrect knowledge and bad experience. This technique lacks ability in dealing with very complex problems, because KBS rely heavily on past experience. The required information may not be available and difficult to be obtained.

Genetic algorithm (GA) is well suited to tackle complex reconfiguration problems, because GA uses information-randomized (genes) exchange to exploit widely available information to tackle different design problems. The major advantage of GA is able to

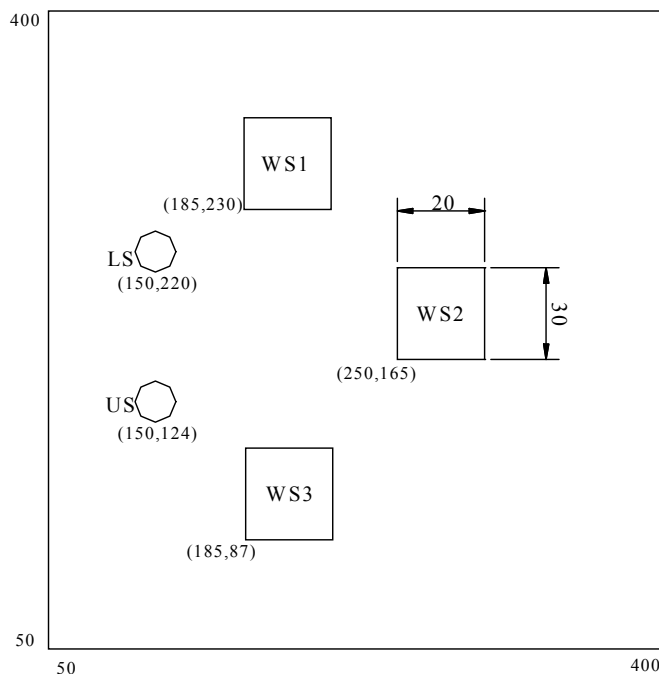
simultaneously provide solutions of satisfying different design objectives. (Man *et al*, 1999).

### 3.3. Genetic algorithm

Genetic algorithm (GA) is a search algorithm using principles of natural evolution. It simulates the evolution of a population of individual based on the rule of the survival of the fittest for the given environment. The pool of chromosomes forming the actual population contains varying genetic material. Selection, crossover, and mutation are major genetic operators. They work selecting pairs of solutions from that population and combining them to produce new solutions. Messy genetic algorithm (MGA), which was introduced by Goldberg *et al*, 1989 and works different from classical GA. MGA works with string of flexible length in which genes can be arranged in any order. The flexible length of MGA could provide the encoding of the diversified properties of the solution to design an assembly line system.

As an example, linear, rotating, and bend conveyors have been selected for the construction of a two-dimensional assembly line system. Three assembly workstations (WS1, WS2 & WS3), loading (LS) & unloading stations (US) have been positioned as shown in Figure 3. The process sequence of the workstations is WS1>WS2>WS3. All processing workstations, loading & unloading stations are in fixed positions. The size and the processing time of each workstation are equal and the length of each linear conveyor is also equal. Table 1 lists the required information of the manufacturing data, control & constraints for the Figure 3.

**Figure 3. The positions of assembly workstations, loading & unloading stations**



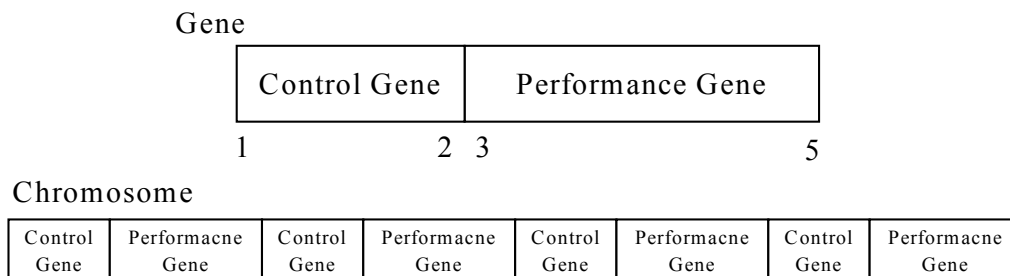
**Table 1. Required information for the reconfiguration of a FALS**

<b>Manufacturing Data</b>	<b>Contents</b>
Number of assembly workstation (WS)	3
Locations of the three WS	As shown in Figure 3.
Locations of the LS & US	As shown in Figure 3.
Processing time of WS	8 minutes (min)
Length of the linear conveyor	3.2 m
Production Volume	6000 units
Time allowed for production	800 hours
<b>Controls</b>	
Transporting time of rotating conveyor	3 seconds (sec)
Transporting time of linear conveyor	0.2 m/sec.
Transporting time of conveyor bend	0.2 m/sec.
The position of a work-part for assembling	As long as the work-part passes closely near to one side of the WS
Orientation of a work-part for assembling	All orientation
<b>Constraints</b>	
Size of the assembly plant	Min_x = 50 Min_y = 50 Max_x = 400 Max_y = 400
Restriction of joining the conveyor-components	Tow rotating conveyors cannot be joined together. All conveyors move in one direction during assembly.

### 3.3.1 Messy coding representation

MGA works with strings of flexible length in which genes are ordered randomly. Each gene is build by a pair of binary sets. First set is control gene that specifies the meaning of the gene (Table 2). Second set is performance gene that represents the executive value of the control gene. Genes are joined together to form a string of values as a chromosome (candidate layout) for evolution.

**Table 2. Sequence of gene layout**



Considering the given example, the linear, rotating, and bend conveyors are joined together in the form of chromosomes. Control gene contains two binary bits (1-2) that

represent the reconfigurable conveyor-components. A total of four different types of conveyor-components can be selected as shown in table 3.

**Table 3. Representations of control gene**

Bit 1-2	Control Gene
00	None
01	Rotating Conveyor
10	Linear Conveyor
11	Conveyor-Bend

Performance gene contains three binary bits that represent the executive value of the control gene. Control gene has the value of “00” that means none of conveyor-components to execute transporting action. If control gene has the value of “01”, the rotating conveyor will perform the actions as tabulated in Table 4.

**Table 4. Actions of performance gene with control gene “01”**

Bit 3	Moving Direction
0	Anti-clockwise
1	Clockwise
Bit 4-5	Rotational Angle
00	0°
01	90°
10	180°
11	270°

If control gene contains “10”, the linear conveyor will perform the actions as tabulated in Table 5. No actions execute on bit 4-5.

**Table 5. Actions of performance gene with control gene “10”**

Bit 3	Moving Direction
0	Backward
1	Forward

If control gene contains “11”, the conveyor-bend will perform the actions as tabulated Table 6.

**Table 6. Actions of performance gene with control gene “11”**

Bit 3	Moving Direction
0	Anti-clockwise
1	Clockwise
Bit 4-5	Rotational Angle
00	30°
01	45°
10	60°
11	90°

### 3.3.2 Decoding

The chromosomes are decoded by binary alphabet into coordinates.

- Decoding of rotating conveyor: let  $x$ ,  $y$  denote the current coordinates,  $x_i$  and  $y_i$  denote target coordinates,  $rad$  denotes radius of the conveyor, and  $\theta$  is the rotational angle of the movement. The target coordinates  $(x_i, y_i)$  will be given by:

$$x_i = x + (rad \cos\theta)$$

$$y_i = y + (rad \sin\theta)$$

- Decoding of linear conveyor: let  $x$ ,  $y$  denote the current coordinates,  $\Delta x$  &  $\Delta y$  denote target coordinates,  $L$  denotes length of linear conveyor and  $\phi$  is the current moving direction of assembly line. The target coordinates  $(x_j, y_j)$  will be given by:

$$x_j = \Delta x + (L \cos\phi)$$

$$y_j = \Delta y + (L \sin\phi)$$

- Decoding of conveyor bend: let  $x$ ,  $y$  denote the current coordinates,  $R$  denotes radius and  $S$  denotes the bend length,  $\theta_b$  is the turning angle of the bend conveyor, and  $\phi$  is the current moving direction of assembly line, The target coordinates  $(x_k, y_k)$  will be given by:

$$S = R\theta_b (\text{radian})$$

$$x_k = x + (R \cos\phi) + (R \cos\theta_b)$$

$$y_k = y + (R \sin\phi) + (R \sin\theta_b)$$

Those decoded chromosomes will be employed in evaluation phase to evaluate the performance of each decoded chromosome (assembly line layout).

### 3.3.3 Evaluation of FALS reconfiguration

The evaluation phase contains two major steps: first is to calculate objective values for each assembly line layout; second is to convert objective values into fitness values.

In the example, two requirements have been defined in separately for reconfiguring different properties of the assembly line system. Requirement 1 is to minimize the number of the reconfigurable conveyor-components. Requirement 2 is to provide three alternative processing paths in the FALS.

- For Requirement 1, four objective functions (*obj\_fun*) have been considered for evaluating each candidate assembly line layout as follow:

1. to evaluate the layout whether the work-part flows from loading to unloading stations;
2. to evaluate the layout whether the work-part passes all processing workstations in sequence;
3. to evaluate the layout whether within the provided space;
4. to evaluate the shortest processing time of the production.

- For Requirement 2, four objective functions (*obj\_fun*) have been considered for evaluating each candidate assembly line layout as follow:

5. to evaluate the layout whether the work-part flows from loading to unloading stations;

6. to evaluate the layout whether the work-part passes all processing workstations in sequence;
7. to evaluate the layout whether within the provided space;
8. to evaluate the maximum numbers of alternative flow paths in the layout.

Each objective function has its own score for representing its objective value. If the layout meets an objective function, the layout will get score '1', otherwise it will get score '0'. Finally, the total objective value will be converted into the fitness value as follow:

$$eval(v_k) = \sum_{o=1}^{obj\_fun} s_o$$

where  $s_o$  is each objective value for each  $obj\_fun$ ;  $eval(v_k)$  is fitness value for each chromosome  $v_k$ .

The evaluation method and fitness calculation of each objective function are described detail in sections: 3.5 & 3.6.

### 3.3.4 Selection

The selection operator is used to select parent for generation. A roulette wheel approach has been adopted as the selection procedures. It belongs to the fitness-proportional selection and can select a new population with respect to the probability distribution based on fitness values. The roulette wheel can be constructed as follows (Man *et al*, 1999):

Step 1: Sum up the fitness value  $eval(v_k)$  for each chromosome  $v_k$ ; named as total fitness (F)

$$F = \sum_{k=1}^{pop\_size} eval(v_k)$$

Step 2: Generate a random number (r) from the range [0, F]

Step 3: Return the first population member whose fitness, added to the fitness of the preceding population member, is greater than or equal to r

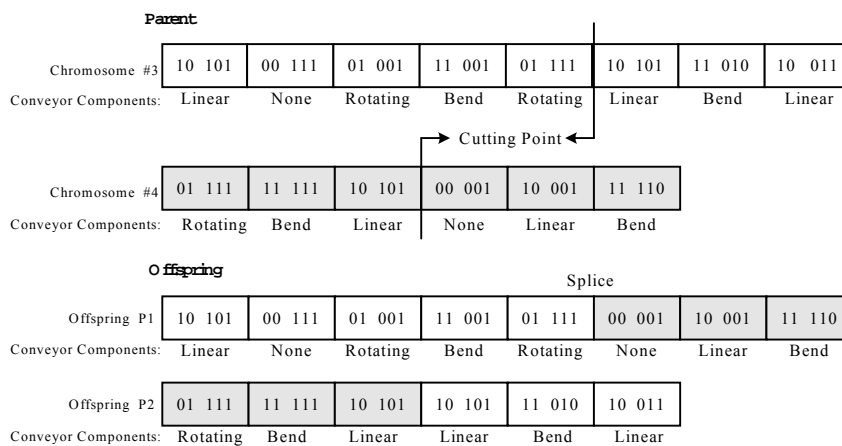
For example, there are four chromosomes in the population pool,  $v_1, v_2, v_3, v_4$  with fitness values 1, 2, 2, 3, respectively. Then calculate cumulative probability ( $q_k$ ),  $q_1=1$ ,  $q_2=1+2=3$ ,  $q_3=1+2+2=5$ ,  $q_4=1+2+2+3=8$ . Now ready to spin the roulette wheel four times, and each time a single chromosome is selected for parent pool. Let us assume that a random sequence of 4 numbers from the range [0, F] is 4, 6, 2, and 7. The first number  $r_1=4$  is greater than  $q_2$  and smaller then  $q_3$ , that means the chromosome  $v_3$  is selected for forming first parent ( $P_1$ ). The second number  $r_2=6$  is greater than  $q_3$  and smaller then  $q_4$ ,

that means the chromosome  $v_4$  is selected for forming second parent ( $P_2$ ); and so on. Finally, the parent pool consists of  $v_3$ ,  $v_4$ ,  $v_2$ , and  $v_4$  chromosomes for crossover.

### 3.3.5 Crossover

Crossover is the main genetic operator. It transfers a portion of genetic codes between two-selected parents and then generates offspring by combining both chromosomes' feature. In MGA the crossover operator is modified. The position of cuts can be chosen independently for both parents. After the cut operation partial strings are spliced in a random order as shown in Figure 4.

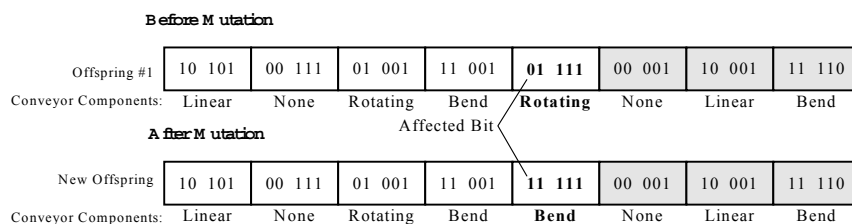
**Figure 4. Schematics of cut and splice operators**



### 3.3.6 Mutation

Mutation is a background operator, which produces spontaneous random changes in various chromosomes. This operator is randomly applied with probability  $P_m$  during evolution and helps to ensure that no point in the search space has a zero probability of being examined (Mitsuo *et al*, 2000). For each gene in a chromosome, an arbitrary choice is made to decide whether the mutation operation is performed or not. If the decision is not to perform the mutation operation, the gene will be kept unchanged. Otherwise, the affected bit may change value from 1 to 0 or from 0 to 1 as shown in Figure 5.

**Figure 5. Schematics of mutation operator**



### 3.3.7 Updating control parameters

The crossover rate is defined as the ratio of the number of offspring produced in each generation to the population size (Mitsuo *et al*, 2000). This ratio controls the expected number of chromosomes to undergo the crossover operation. In order to improve efficiency of the evolutionary process, the  $P_c$  should be updated during the fitness average ( $f_{av}$ ) of population  $\geq$  Max. Fitness Value ( $F_{max}$ ) - 1, that means when the  $f_{av}$  is greater or equal to  $F - 1$ , the  $P_c$  will be changed from 0.9 to 0.65. Because a high crossover rate allows exploration of more of the solution space and reduces the chances of settling for a

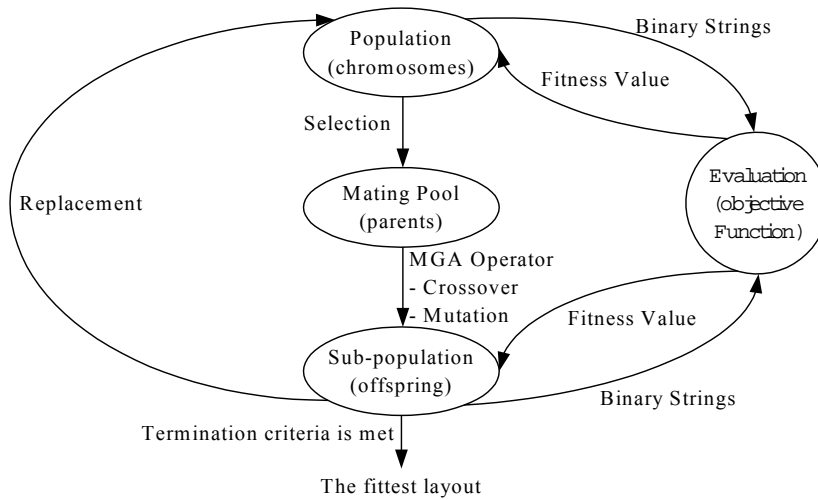
false optimum; but if this rate is too high, it results in the wastage of a lot of computation time in exploring unpromising regions of the solution space.

### 3.4. Implementation of the MGA approach

Let  $pop\_size$  denotes the size of population, let  $P_c$  denotes the crossover rate, let  $P_m$  denotes the mutation rate, and let  $max\_gen$  denotes the maximal generation for a run. The data flow chart of the MGA cycle is illustrated in Figure 6., and the procedures of operating MGA are summarized as follows:

- Step 0 (parameter setting): Set evolutionary environment: workstations, loading and unloading positions,  $pop\_size = 60$ ,  $P_c = 0.9$  for  $f_{av} < F_{max} - 1$  and  $P_c = 0.65$  for  $f_{av} \geq F_{max} - 1$ ,  $P_m = 0.001$ , and  $max\_gen = 200$ .
- Step 1 (initialization): Randomly generate initial population containing  $pop\_size$  chromosomes.
- Step 2 (evaluation): Decode the chromosomes and calculate fitness value of each layout.
- Step 3 (selection): Make a roulette wheel selection to select fitter chromosomes from the current population for generation.
- Step 4 (crossover): Make  $pop\_size \times P_c$  offspring using the cut and splice operators.
- Step 5 (mutation): Make  $off\_size \times P_m$  offspring using the proposed mutation operator.
- Step 7 (evaluation): The offspring have been generated and evaluate their fitness values. If generation equal to the  $max\_gen$  or the fittest layout has been generated, stop the evolutionary process; otherwise go to step 8.
- Step 8 (replacement): Insert the new chromosomes into a new population and go back to step 3.

**Figure 6. MGA cycle**



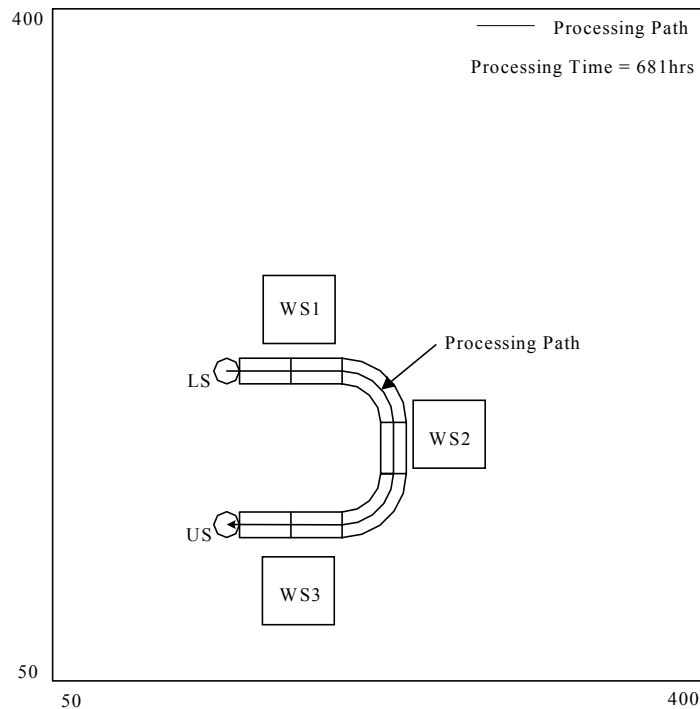
### 3.5. Fulfillment of requirement 1

The proposed method has been applied for the minimization of the number of reconfigurable conveyor-components. The possible layout for this case is given in figure 7. It has fulfilled the four objective functions. The fitness value of this layout is 4. The calculation of evaluation phase is listed in table 7.

**Table 7. Evaluation of the objective functions for requirement 1**

<b>Decoded Chromosome</b> (Requirement 1)	(150,220) (182,220) (214,220) (246,180) (246,156) (214,124) (182,124) (150,124)	
<b>Objective Functions</b>	<b>Evaluation Criteria</b>	<b>Objective Value (<math>s_o</math>)</b>
1. Does the work-part flow from LS to US?	If the first and the last coordinates of decoded chromosome of the layout are equal to the given coordinates as shown in Figure 3, the $s_1$ of the objective function will get 1, otherwise will get 0.	1
2. Does the work-part pass three WS in sequence?	If the distance between the conveyor and a WS is smaller than 30mm, that means the conveyor has passed the WS, the $s_2$ of the objective function will get 1, otherwise will get 0.	1
3. Is the layout smaller than the given space as shown in Table 1?	If the overall size of the layout is smaller than the given space as shown Table 1, the $s_3$ of the objective function will get 1, otherwise will get 0.	1
4. to minimize the processing time	The processing time is calculated by: $[(N_R \times t_R) + (N_L \times t_L) + (N_B \times t_B)] / P_{vol}$ where, $N_R, N_L, N_B$ is the number of rotating, linear, bend conveyors.	1
<b>Fitness Value</b> $eval(v)$	$eval(v_k) = \sum_{o=1}^{obj\_fun} s_o$	4

**Figure 7. The best of assembly line layout of meeting requirement 1**



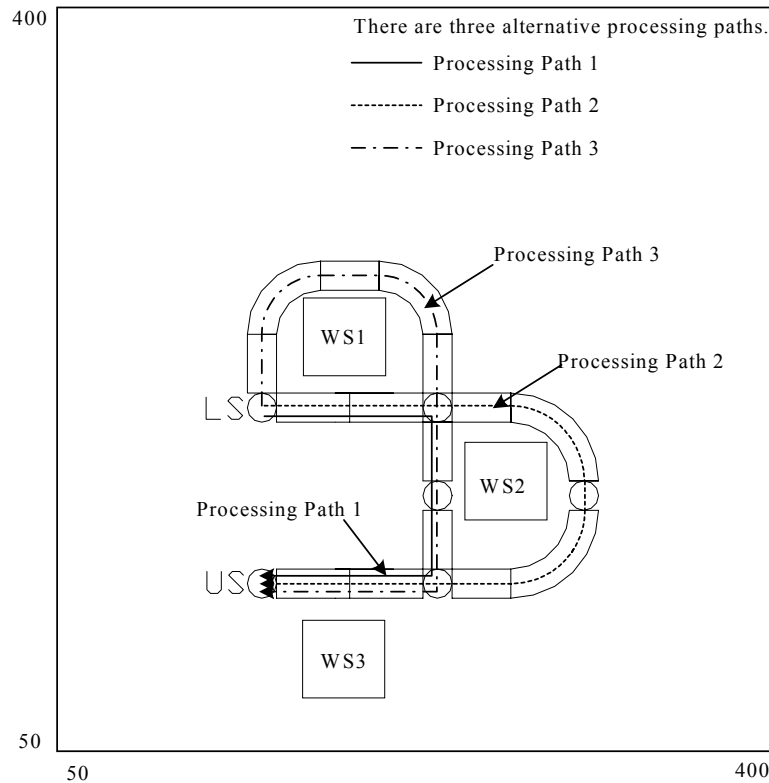
### 3.6. Fulfillment of requirement 2

The proposed method has also been applied to provide three alternative processing paths. The possible layout for this case is given in figure 8. It has fulfilled the four objective functions. The fitness value of this layout is 4. The evaluation methods for the first three objective functions same as requirement 1. The calculation of evaluation phase is listed in table 8.

Table 8. Evaluation of the objective functions for requirement 2

<b>Decoded Chromosome</b> (Requirement 2)	(150,220) (150,252) (182,284) (214,284) (246,252) (246,220) (214,220) (182,220) (214,220) (246,220) (278,188) (246,188) (246,220) (246,188) (246,156) (246,124) (214,124) (182,124) (150,124)	
<b>Objective Functions</b>	<b>Evaluation Criteria</b>	<b>Objective Value (<math>s_o</math>)</b>
1 to 3	Same as requirement 1.	3
4. to evaluate the number of alternative paths	If the layout has three alternative flow paths, the $s_4$ of the objective function will get 1, otherwise will get 0.	1
<b>Fitness Value</b> $eval(v)$	$eval(v_k) = \sum_{o=1}^{obj\_fun} s_o$	4

**Figure 8. The best of assembly line layout of meeting requirement 2**



#### **4. DISCUSSION**

The proposed GA has been tackled two different requirements. The FALS of fulfilling the requirement 1 is inflexible to cope with unplanned events occurred during the operation such as system component breakdown or suddenly call for product change. The operation has to stop. As the result, the production cost and time may be increased.

The FALS of fulfilling the requirement 2, three alternative processing paths have been generated in the layout. The alternative paths pass through the each process workstations in a given sequence. Although the duplication of conveyor-components has increased the production cost, the FALS is able to deal with the change of production requirements and unplanned events occurred during assembly.

#### **5. CONCLUSION**

The proposed GA is able to reconfigure a FALS to meet the desire production requirements. The proposed approach offers a method for the selection amount all possible FALS configurations of satisfying given production requirements. The advantage of the proposed GA doesn't rely on the past experience to reconfigure the FALS.

#### **ACKNOWLEDGEMENTS**

The authors would like to express their appreciation to City University of Hong Kong to support this research (project no.: 7001295-630).

## REFERENCES

- Azadivar, F. & Wang, J.J. (2000) Facility layout optimization using simulation and genetic algorithms. International Journal of Production Research, 38 (17) November, pp. 4369-4383.
- Goldberg, D.E. & Korb, B. Deb, K. (1989) Messy genetic algorithms: motivation, analysis, and first results. Complex Systems, 3, pp.349-530.
- Goldberg, D.E. Deb, K. & Korb, B. (1990) Messy genetic algorithms revisited: studies in mixed size and scale. Complex Systems, 4, pp.415-444.
- Ho, John K.L. & Ranky, Paul G. (1997) Object oriented modeling and design of reconfigurable conveyors in flexible assembly systems. International Journal of Computer Integrated Manufacturing, 10 (5) September, pp.360-379
- Lee, M.K. Luong, H.S. & Abhary, K. (1997) A genetic algorithm based cell design considering alternative routing. Computer Integrated Manufacturing System, 10 (2) May, pp.93-107.
- Leung, M.K. (2000) Research into reconfigurable assembly line system. Mphil. thesis, City University of Hong Kong.
- Lindkvist, R.G.T. (1985) Handbook of materials handling / compiled by the Transportforskningskommissionen. New York Chichester, E. Horwood.
- Malki, O. (2001) Genetic Algorithms and Genetic Programming [Internet], Institute for Applied Physics University of Kiel Olshausenstr. Available from: <http://www.ang-physik.uni-kiel.de/~oliver/gagp.html> [Accessed 01 March, 2001].
- Man, K.F. Wong, Y.S. & Kwong, S. (1999) Genetic algorithms concepts and designs. Hong Kong, Springer.
- Mitsuo, G. & Cheng, R. (2000) Genetic algorithms and engineering optimization. New York, Wiley.
- Myung, S. & Han, S. (2001) Knowledge-based parametric design of mechanical products based on configuration design method. Expert Systems with Applications, 21(2) August, pp.99-107.
- Zha, X.F. Du, H. & Lim, Y.E. (2001) Knowledge intensive Petri net framework for concurrent intelligent design of automatic assembly lines. Robotic and Computer Integrated Manufacturing, 17(5) October, pp.379-398.