Int J Adv Manuf Technol (2008) 38:143–153
DOI 10.1007/s00170-007-1069-x

ORIGINAL ARTICLE

Optimization of flexible process planning by genetic programming

X. Y. Li · X. Y. Shao · L. Gao

Received: 15 November 2006 / Accepted: 27 April 2007 / Published online: 31 May 2007

Abstract The traditional manufacturing system research literature generally assumed that there was only one feasible process plan for each job. This implied that there was no flexibility considered in the process plan. But, in the modern manufacturing system, most jobs may have a large number of flexible process plans. So, flexible process plans selection in a manufacturing environment has become a crucial problem. In this paper, a new method using an evolutionary algorithm, called genetic programming (GP), is presented to optimize flexible process planning. The flexible process plans and the mathematical model of flexible process planning have been described, and a network representation is adopted to describe the flexibility of process plans. To satisfy GP, it is very important to convert the network to a tree. The efficient genetic representations and operator schemes also have been considered. Case studies have been used to test the algorithm, and the comparison has been made for this approach and genetic algorithm (GA), which is another popular evolutionary approach to indicate the adaptability and superiority of the GP-based approach. The experimental results show that the proposed method ispromising and very effective in the optimization research of flexible process planning.

Keywords Flexible process planning · Process plans selection · Genetic programming (GP) · Optimization

X. Y. Li · X. Y. Shao · L. Gao (✉)
The State Key Laboratory of Digital Manufacturing Equipment and Technology,
Huazhong University of Science and Technology,
Wuhan, Hubei, China
e-mail: gaoliang@mail.hust.edu.cn

1 Introduction

A process plan specifies what raw materials or components are needed to produce a product, and what processes and operations are necessary to transform those raw materials into the final product. It is the bridge between product design and manufacturing. The outcome of process planning is the information for manufacturing processes and their parameters, and the identification of the machines tools, and fixtures required to perform those processes.

Generally, the traditional manufacturing system research literature assumed that there was only one feasible process plan for each job. This implied that there was no flexibility possible in the process plan. But, in the modern manufacturing system, most jobs may have a large number of flexible process plans. So, flexible process plan selection in a manufacturing environment has become a crucial problem. Because it has a vital impact on manufacturing system performance, several researchers have examined the flexible process plans selection problem in recent years. Sormaz and Khoshnevis [1] describe a methodology for generation of alternative process plans in the integrated manufacturing environment. This procedure includes selection of alternative machining processed, clustering and sequencing of machining processes, and generation of a process plan network. Kusiak and Finke [2] developed a model to select a set of process plans with minimum cost of removing material and minimum number of machine tools and other equipments. Bhaskaran and Kumar [3] formalized the selection of process plans with the objective of minimizing the total processing time and the total steps of processing. Lee and Huy [4] presented a new methodology for flexible operation planning using the Petri net which was used as a unified framework for both operation planning and plan representation. Ranaweera and Kamal [5] presented a

technique for evaluating processing plans generated by a cooperative intelligent image analysis framework, and this system was able to rank multiple processing plans. Seo and Egbelu [6] used tabu search to select a plan based on product mix and production volume. Usher and John [7] used genetic algorithms to determine optimal, or near-optimal, operation sequences for parts of varying complexity. Tiwari [8] used genetic algorithm to obtain a set of process plans for a given set of parts and production volume. Rocha and Ramos [9] used genetic algorithm approach to generate the sequence of operations and to select the machine and tools that minimize some criteria. Dereli and Filiz [10] introduced the GA-based optimization modules of a process planning system called optimized process planning system for prismatic parts (OPPS-PRI). Most of these approaches proposed the models to optimize flexible process plans. Moreover, few of them used evolutionary algorithms, and none of them used genetic programming. But evolutionary algorithm is becoming a useful, promising method for solving complex and dynamic problems [11]. This paper presents a new methodology which uses genetic programming that can optimize flexible process planning effectively.

GP is one of the evolutionary algorithms (EA) [12]. In GP, a computer program is often represented as a tree (a program tree) [13], where the internal nodes correspond to a set of functions used in the program and the external nodes (terminals) indicate variables and constants used as the input of functions. Manufacturing optimization has been a major application field for evolutionary computation methods [14]. But it has rarely been the subject of genetic programming research [15, 16]. One of the possible reasons for the lack of GP applications in manufacturing optimization is the difficulty of evolving a direct permutation through GP. Now a new methodology which uses genetic programming that can effectively optimize flexible process plans.

The remainder of this paper is organized as follows. Section 2 introduces flexible process planning. GP is briefly reviewed in Sect. 3. GP for flexible process planning is described in Sect. 4. Case studies and discussion are reported in Sect. 5. The last section is the conclusion.

2 Flexible process planning

2.1 Flexible process plans

There are three types of flexibility considered in flexible process planning [17] [18]: operation flexibility, sequencing flexibility and processing flexibility [19]. Operation flexibility [20], which is also called routing flexibility [21], relates to the possibility of performing one operation on alternative machines, with possibly distinct processing time and cost. Sequencing flexibility is decided by the possibility of interchanging the sequence of the required operations. Processing flexibility is determined by the possibility of processing the same manufacturing feature with alternative operations or sequences of operations. Better performance in some criteria (e.g., production time) can be obtained by the consideration of these flexibilities [20].

Figure 1 shows an example part which consists of three manufacturing features. And the technical specifications for the part have been defined in Table 1. This part has three types of flexibility. From the Table 1, it can be found that every operation can be processed on alternative machines with distinct process time (Oper1 can be processed on M1 and M2 with different processing time), the manufacturing sequence of feature 1 and feature 3 can be interchanged (Oper1 and Oper10 can be interchanged), and in the second column of Table 1, every feature has alternative operations (feature 1 has 4 alternative operations, feature 2 has 4 alternative operations, and feature 3 has 3 alternative operations).

2.2 Representation of flexible process plans

There are many methods used to describe the three types of flexibility [22], such as Petri-net [4], AND/OR graphs and network. And, a network representation proposed by Sormaz [1], Kim [20] and Ho [23] is used here. There are three node types in the network: starting node, intermediate node and ending node [20]. The starting node and the ending node, which are dummy ones, indicate the start and the end of the manufacturing process of a job. An intermediate node represents an operation, which contains the alternative machines that can perform the operation and the processing time required for the operation according to the machines. The arrows connecting the nodes represent the precedence

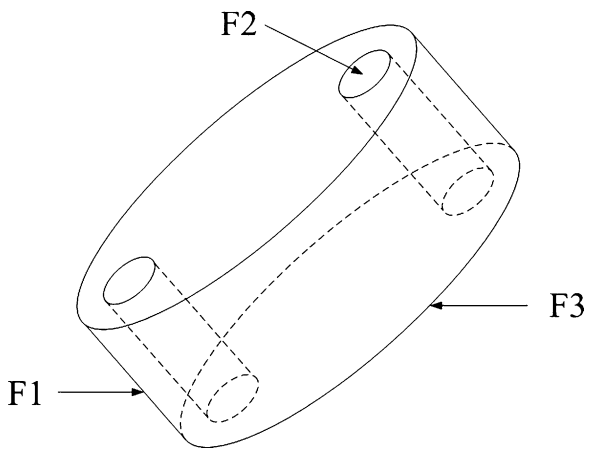

Fig. 1 The example part

Table 1 The technical specifications for the part

Features	Alternative operations	Alternative machines	Working time for each alternative machine (s)
F1	Turning (Oper1)	M1, M2	41, 38
F1	Turning (Oper11)	M3, M4	92, 96
F1	Turning (Oper12)	M5, M6	20, 23
	Fine turning (Oper13)	M1, M2	65, 70
F1	Turning (Oper12)	M5, M6	20, 23
	Grinding (Oper14)	M7, M9	68, 72
F2	Drilling (Oper3)	M2, M4	20, 22
	Reaming (Oper4)	M1, M2, M5	35, 29, 36
	Boring (Oper9)	M2, M3, M4	50, 45, 50
F2	Drilling (Oper6)	M2, M3, M4	25, 20, 27
	Reaming (Oper7)	M7, M8	54, 50
	Boring (Oper9)	M2, M3, M4	50, 45, 50
F2	Reaming (Oper8)	M5, M6	80, 76
	Boring (Oper9)	M2, M3, M4	50, 45, 50
F2	Reaming (Oper15)	M7, M8, M9	50, 56, 52
F3	Turning (Oper2)	M5, M7	75, 70
F3	Milling (Oper5)	M9, M10	49, 47
F3	Milling (Oper10)	M9, M10	70, 73

between them. OR relationships are used to describe the processing flexibility that the same manufacturing feature can be performed by different process procedures. If the links following a node are connected by an OR connector, it only need to traverse one of the OR-links (the links connected by the OR-connector are called OR-links). OR-link path is an operation path that begins at an OR-link and ends as it merges with the other paths, and its end is denoted by a JOIN-connector. For the links that are not connected by OR-connectors, all of them must be visited [20]. Based on the technical specifications and precedence constraints, the flexible process plans of the part can be converted to the network.

Figure 2 shows the example part's (see Fig. 1) flexible process plans network which is converted from the technical specifications shown in Table 1, and this network will be used in Sect. 5. In this network, paths {11}, {12, 13} and {12, 14} are three OR-link paths. An OR-link path can of course contain the other OR-link paths, e.g., paths {6, 7} and {8}.

2.3 Mathematical model of flexible process planning

In this paper, the optimization objective of the flexible process planning problem is to minimize the production time (contains working time and transmission time).

In solving this problem, the following assumptions are made [20]:

(1) Each machine can handle only one job at a time.
(2) All machines are available at time zero.
(3) After a job is processed on a machine, it is immediately transported to the next machine on its process, and the transmission time among machines is constant.

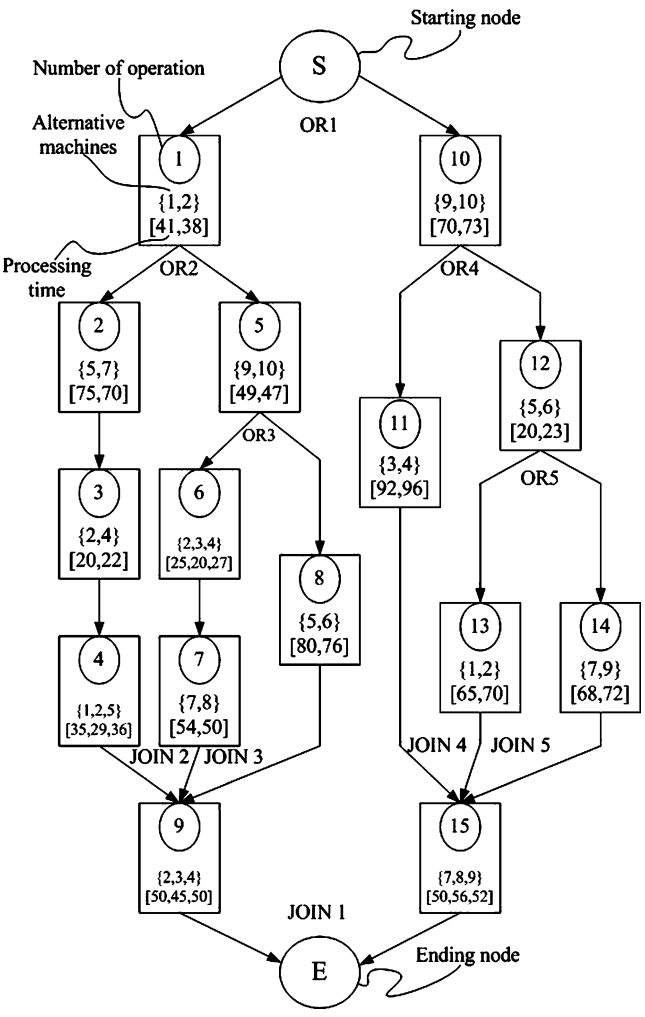

Fig. 2 Flexible process plans network

(4) The different operations of one job can not be processed simultaneously.

Based on these assumptions, the mathematical model of flexible process planning is described as follows:

The notations used to explain the model are described below:

N	the total number of jobs;
G_I	the total number of flexible process plans of the ith job;
o_{ijl}	the jth operation in the lth flexible process plan of the ith job;
P_{il}	the number of operations in the lth flexible process plan of the ith job;
k	the alternative machine corresponding to o_{ijl};
$TW(i, j, l, k)$	the working time of operation o_{ijl} on the kth alternative machine;
$TS(i, j, l, k)$	the starting time of operation o_{ij} on the kth alternative machine;
$TT(i, l, (j, k_1), (j + 1, k_2))$	the transmission time between the k_1th alternative machine of the o_{ij} and the k_2th alternative machine of the $o_{i(j+1)l}$;
$TP(i)$	the production time of the ith job;

The objective function is

$$\min \quad TP(i) = \sum_{j=1}^{P_{il}} TW(i,j,l,k) + \sum_{j=1}^{P_{il}-1} TT(i,l,(j,k_1),(j+1,k_2)) \quad i \in [1,N], j \in [1,P_{il}], l \in [1,G_i] \tag{1}$$

Each machine can handle only one job at a time. This is the constraint of machine.

$$TS(i,j_2,l,k) - TS(i,j_1,l,k) > TW(i,j_1,l,k) \quad i \in [1,N], j_1, j_2 \in [1,P_{il}], l \in [1,G_i] \tag{2}$$

The different operations of one job cannot be processed simultaneously. This is the constraint of different processes for one job.

$$TS(i,(j+1),l,k_2) - TS(i,j,l,k_1) > TW(i,j,l,k_1) \quad i \in [1,N], j \in [1,P_{il}], l \in [1,G_i] \tag{3}$$

The objective function is Eq. (1), and the two constraints are in Eqs. (2) and (3).

3 Brief review of GP

Genetic programming (GP) was introduced by Koza [24, 25] as a method for using natural selection and genetics as a basis for automatically creating computer programs.

For a given problem, the work steps of GP are then given as follows:

Step 1: Initialize population randomly generated computer programs (trees).

Step 2: Evaluate all population.

Step 3: Produce a new generation population:

(1) Reproduction
Reproduce some excellent individuals and delete the same number of inferior individuals.

(2) Crossover
According to the user-defined probabilistic, some individuals are selected to be crossovered. For each two selected trees in a pair, a crossover point is chosen randomly and two offspring (trees) are produced from the pair in terms of the crossover operation and are placed into the new generation.

(3) Mutation
According to the user-defined probabilistic, some individuals are selected to be mutated. For each selected tree, a mutation point is randomly chosen, and one offspring (tree) is produced from the selected one in terms of the mutation operation and is placed into the new generation.

Step 4: Do steps 2 and 3 cyclically until terminating condition satisfied.

There are a number of issues to be considered in a GP system [12]:

(1) Definitions of functions and terminals to be used in the trees generated.
(2) Definition of a fitness function for evaluating trees and the way those trees are evaluated.
(3) Generation of the initial population.
(4) Selection strategies for trees to be included in next generation population.
(5) How reproduction, crossover and mutation operations are carried out and how often these operations are performed.
(6) Criteria for terminating the evolution process and the way to check if the terminating conditions are satisfied.
(7) Return of the final results.

4 GP for flexible process planning

Using GP for flexible process planning has some advantages. GP provides a mathematical representation of the

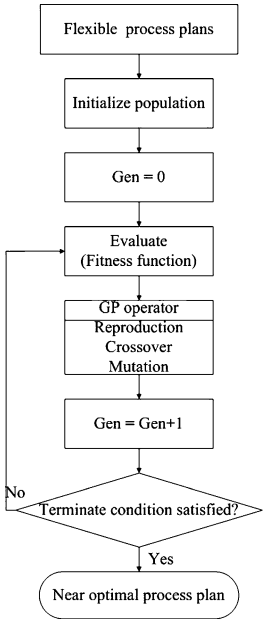

Fig. 3 Flow chart of GP

flexible process plans. Now, it is described that how GP can be used to optimize flexible process planning.

4.1 The flow chart of proposed method

Figure 3 shows the flow chart of the proposed method (GP for flexible process planning). First, CAPP system gives the flexible process plans. And then, the search begins with an initial population. The individual consists of two parts. One part is represented by the sequence of operations and the set of machines used to accomplish the operation, the other one is composed by discrimination value. The detailed description of individual will be given in Sect. 4.2. The rest steps of the method are the same as the common GP.

4.2 Convert network to tree, encoding and decoding

4.2.1 Convert network to tree

From Fig. 2, it is known that flexible process plans can be represented as a network. And in GP, the individual is often represented as a tree (see Sect. 3). So, the key of the proposed method is how to convert network to tree.

In order to convert network to tree, a method has been presented. The first step of this method is deleting the ending-nodeof the network, and the second step is disentwining JOIN-connector. The last step is adding the latter intermediate nodes which are linked by the JOIN-connector to the end point of each OR-link linked by the JOIN-connector. And then, the network has been converted to tree.

A part network of job (see Fig. 2) has been taken as an example to explain how to convert network to tree (see Fig. 4). The procedure is as follows:

Step 1: Delete the ending-node.
Step 2: Disentwine JOIN2-connector and JOIN3-connector.
Step 3: Add operation 9 (the latter intermediate node is linked by the JOIN2-connector and JOIN3-connector) to the end point of path {3, 4}, {6, 7} and {8} (the OR-links which are linked by the JOIN2-connector and JOIN3-connector) respectively.

4.2.2 Encoding and decoding

GP uses the tree hierarchy frame to express problems. Each tree within a member produces one output. A tree which is made up of nodes can be classified to two sets: the function set and the terminal set. The function node is the method,

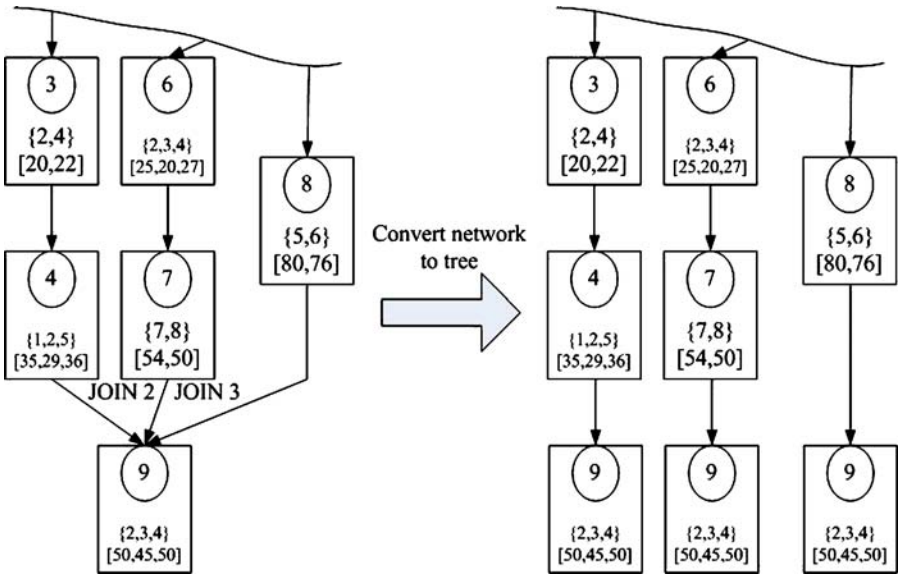

Fig. 4 How to convert network to tree

and the terminal node is the value of the problem. Each node has zero or more inputs and uses those inputs to create its output. A node can have any number of inputs. The terminal can also be thought of as zero-argument function. Input features and any constants are represented by terminal nodes. A node with one or more inputs is a function; its output is dependent on its inputs. For instance, addition, subtraction, multiplication and division all are functions.

In this paper, each tree of each individual is generated by the function set F= {switch-case, link} and terminal set T= {discrimination value, gene}. Switch-case is the conditional expression; and link, which links the nodes together, is a user-defined function, and its output is a list. It includes the nodes which are linked by this function. The sequence of the string is from top to bottom. The discrimination value encodes OR-connectors as the decimal integer. It is in concert with the switch-case function to decide which OR-link will be chosen. A gene is a structure and made up of two parts. The first number is the operation. It can be all the operations of a job, even those may not be performed because of alternative operation procedures. The second one is alternative machine. It is the *i*th element of which represents the machine on which the operation corresponding to the *i*th element of part I is processed. The encoding scheme of a tree is a list that has two parts: part I is made up of genes, and part II is made up of discrimination values.

Figure 5 shows an example individual of job (see Fig. 1). Taking gene (2, 5) for example, 2 is the operation of the job, and 5 is the alternative machine, which corresponds to the operation 2. The encoding scheme of this individual is shown in Fig. 5. Part I is made up of 19 genes; part II is made up of five discrimination values.

The encoding is directly decoded. The selection of the OR-link paths which contain operations and the corresponding machines is decided by the interpretation of part II of the individuals' encoding scheme. And then the orders appearing in the resulting part I are interpreted as an operation sequence and the corresponding machining sequence for the job. In the above encoding example, the operation sequence together with the corresponding machining sequence is (1, 1)-(5, 9)-(6, 3)-(7, 8)-(9, 2).

4.3 Initial population and fitness evaluation

4.3.1 Initial population

In order to operate evolutionary algorithm, an initial population is needed. The generation of the initial population

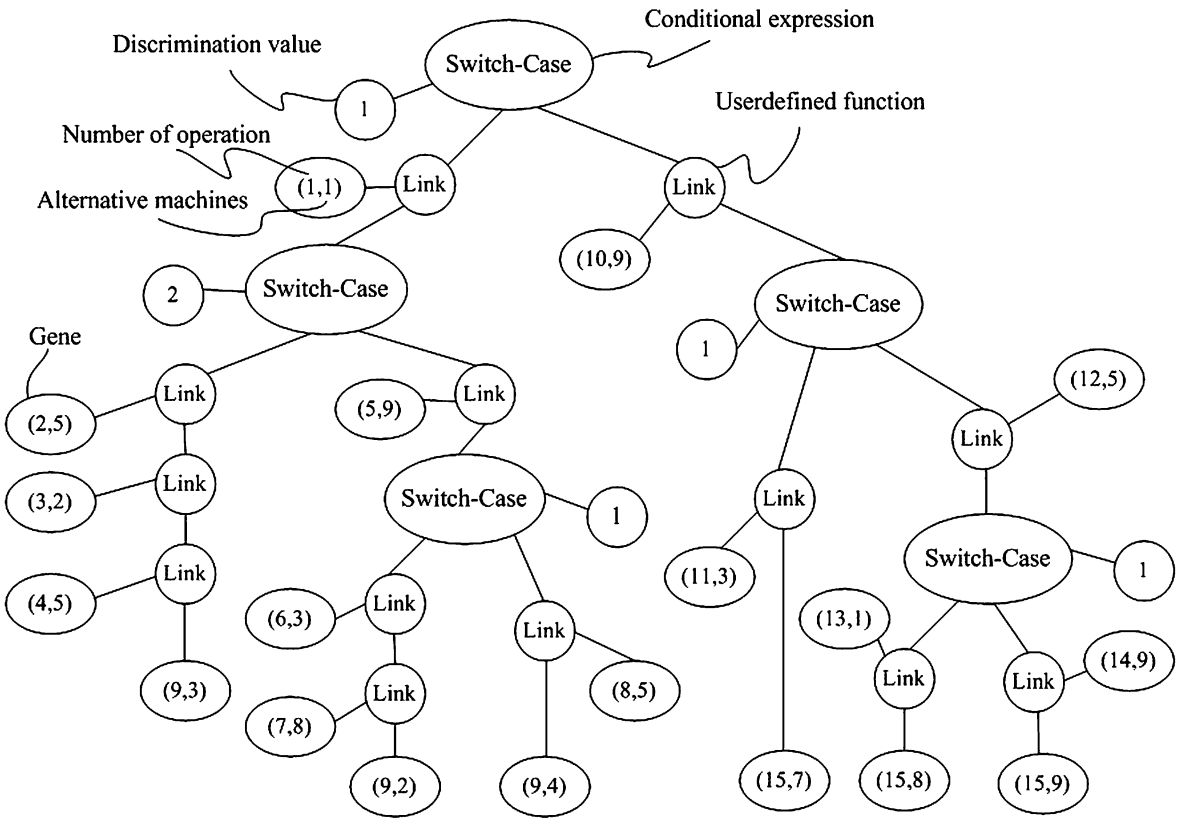

Fig. 5 A tree of individual

in GP is usually done randomly. But when generating the individuals for an initial population of flexible process planning, feasible operation sequence in a process plan has to be taken into account. Feasible operation sequence means that the order of elements in the used encoding does not break constraints on precedence relations of operations [20]. As mentioned above, a method was proposed to generate a random and feasible individual.

The procedure of the method is as follows:

Step 1: The part I of the initial individual contains all the alternative operations, and the sequence of operations is fixed.

Step 2: The second number of part I is created by randomly assigning a machine in the set of machines which can perform the operation placed at the corresponding position in part I.

Step 3: The part II of the initial individual, which represents OR-link paths, is initiated by randomly generating a decimal integer for each component of this part. The selection area of each discrimination value is decided by the number of OR-link paths which are controlled by this value. For example, if it has three OR-link paths, the selection area of the discrimination value is the random decimal integer in [1, 3].

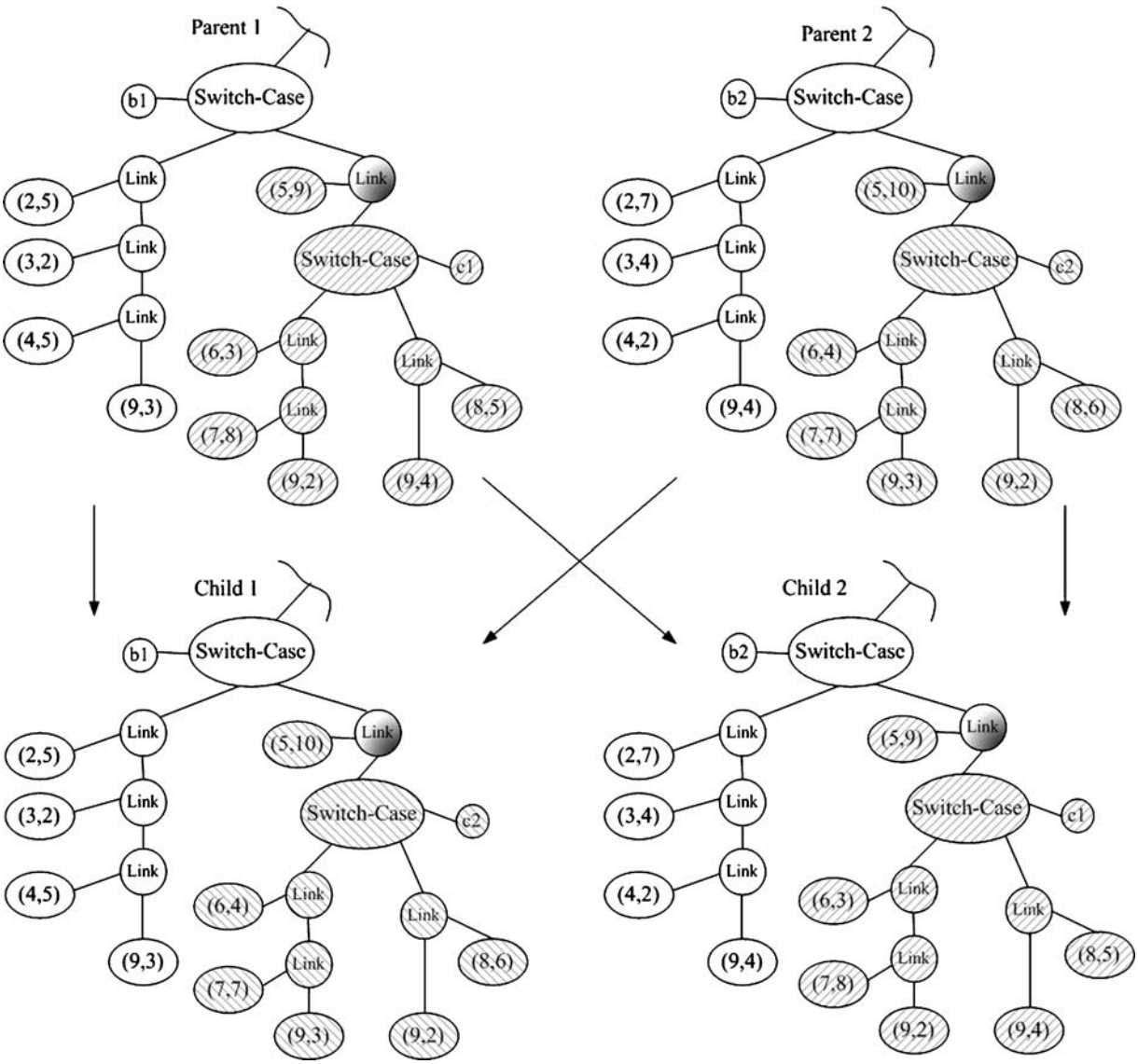

Fig. 6 Subtree exchange crossover

4.3.2 Fitness evaluation

The objective of the flexible process planning problem is to minimize the production time (contains working time and transmission time) for the given problem. Adjusted fitness has been used as the objective. It can be calculated from the following:

$$\max \quad f(i,t) = \frac{1}{TP(i,t)} \tag{4}$$

S	the size of population;
M	the maximal generation;
t	1, 2, 3,…M generations;
$TP(i, t)$	the production time of ith job in the tth generation (see Eq. (1));

The fitness function is calculated for each individual in the population as described in Eq. (4).

4.4 GP operators

It is important to employ good operators that can effectively deal with the problem and efficiently lead to excellent individuals residing in the population. The GP operators can generally be divided into three classes: reproduction, crossover and mutation. And in each class, a large number of operators have been developed [26].

4.4.1 Reproduction

Tournament selection scheme with a user-defined reproduction probabilistic was used for reproduction operation. In tournament selection, a number of individuals are selected at random (dependent on the tournament size, typically between 2 and 7) from the population and the individual with the best fitness is chosen for reproduction. The tournament selection approach allows a tradeoff to be made between exploration and exploitation of the gene pool [26]. This scheme can modify the selection pressure by changing the tournament size.

This scheme has two working steps:

Step 1: Select user-defined tournament size individuals from the population randomly to compose a group.

Step 2: Copy the best member of the group (the one with the best fitness value) to the following generation, and then applying the tournament selection scheme to the remaining individuals.

4.4.2 Crossover

Subtree exchange crossover has been used as the crossover operator here, and fitness-proportion selection scheme with a user-defined crossover probabilistic was used for crossover operation. Subtree exchange crossover can generate feasible children individuals that satisfy precedence restrictions and avoid duplication or omission of operations as follows. The cut point is chosen randomly in the tree, and the subtree before the cut point in one parent (parent 1) is passed on to the same position as in the offspring (child 1). The other part of the offspring (child 1) is made up of the subtree after the cut point in the other parent (parent 2). The other offspring (child 2) is made up of the subtree before the cut point in one parent (parent 2) and the subtree after the cut point in the other parent (parent 1). An example of the crossover is presented in Fig. 6. The cut point is marked with "●". The crossover operator produces feasible trees since both parents are feasible and offspring are created without violating the feasibility of the parents.

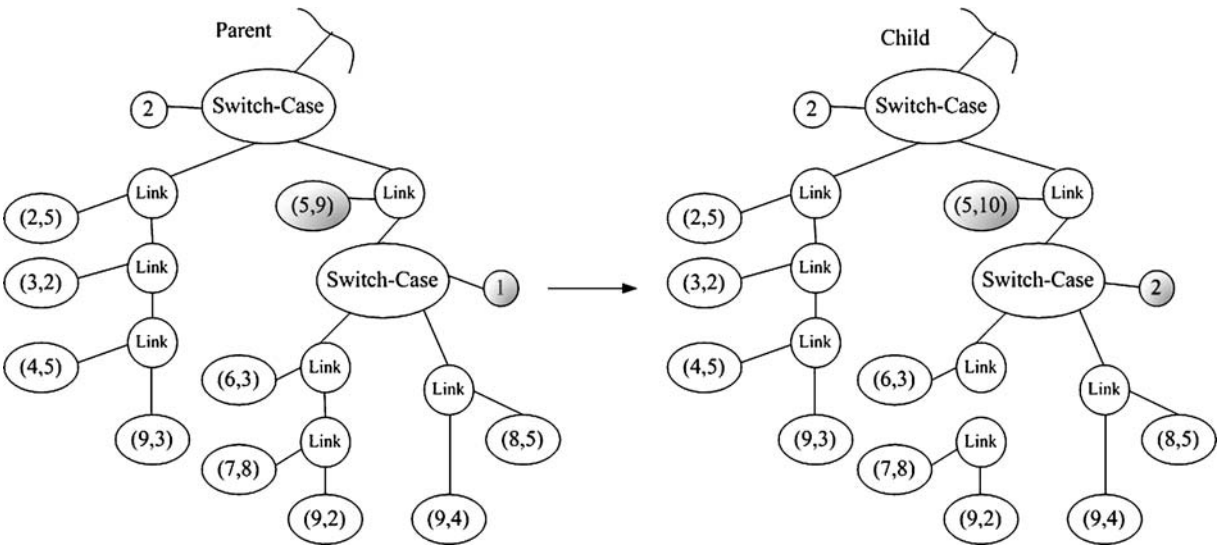

Fig. 7 Point mutation

Table 2 The transmission time between the machines

Machine code	1	2	3	4	5	6	7	8	9	10
1	0	5	8	12	15	4	6	10	13	18
2	5	0	3	7	10	6	4	6	10	13
3	8	3	0	4	7	10	6	4	6	10
4	12	7	4	0	3	14	10	6	4	6
5	15	10	7	3	0	18	12	10	6	4
6	4	6	10	14	18	0	5	8	12	15
7	6	4	6	10	12	5	0	3	7	10
8	10	6	4	6	10	8	3	0	4	8
9	13	10	6	4	6	12	7	4	0	4
10	18	13	10	6	4	15	10	8	4	0

4.4.3 Mutation

Point mutation has been used as the mutation operator here, and random selection scheme with a user-defined mutation probabilistic was used for mutation operation. Each of the selected individuals is mutated as follows. First, the point mutation scheme is applied in order to change the alternative machine represented in the gene (see Fig. 5) of tree. A gene is randomly chosen from the selected individual. Then, the second element of gene is mutated by altering the machine number to another one of the alternative machines at random. Second, the other mutation is carried out to alter the OR-link path. This is associated with part II of encoding scheme of tree. A discrimination value is randomly chosen from the selected individual. Then, it is mutated by changing its value in the selection area randomly. In the example depicted in Fig. 7, mutation point is marked with "●". Gene (5, 9) has changed into (5, 10), and the selected discrimination value has changed from 1 to 2.

5 Case studies and discussion

Some experiments have been conducted to measure the adaptability and superiority of the proposed GP approach.

Table 3 The GP and GA parameters

Parameters	GP		GA	
	Job 1	Job 2	Job 1	Job 2
The size of the population, S	400	400	400	400
Total number of generations, M	30	30	60	60
Probability of reproduction operation, p_r	0.05	0.05	0.05	0.05
Probability of crossover operation, p_c	0.50	0.50	0.50	0.50
Probability of mutation operation, p_m	0.05	0.05	0.05	0.05
Tournament size, b	2	2	2	2

Table 4 Experiment results

Job	BIF	MPAF	CPU time (s)
1	0.00467289	0.00444361	113.6
2	0.00444444	0.00437222	130.9

And, the algorithm has been compare with genetic algorithm (GA), which is another popular heuristic algorithm. The performance of the approach is satisfactory from the experiments and comparison.

5.1 Implementation and testing

For doing the experiments of the proposed approach, two jobs with flexible process plans have been generated. Job 1 has been given in Figs. 1 and 2, and job 2 is changed from job 1 by assuming the second machine is broken in the current shop status. It has ten machines on the shop floor. The code of the machine in job 2 is the same as the code of machine in job 1. The transmission time (the time units is the same as processing time in Fig. 2) between the machines is given in Table 2. The objective is to solve the optimization of flexible process plans with the maximum objective function $f(i, t)$ (Eq. (4)). The GP parameters for the two jobs are given in Table 3. The terminating condition is reaching the maximum generation. The GP is coded in C++, and implemented on a PC (Pentium (R) 4, CPU 2.40 GHz).

The experiments are carried out for the objective: minimizing the production time (see Sect. 4.3.2). The experimental results (fitness is the adjusted fitness) of the 2 jobs, which include the best individual's fitness (BIF), the maximum population's average fitness (MPAF), and CPU time are reported in the Table 4.

From the experimental results which are shown in Table 4, the best process plan of each job has been shown in Table 5. And, Fig. 8 illustrates convergence curves of GP for the 2 jobs. The curves show the search capability and evolution speed of this algorithm.

From above experiment results which are shown in Table 5, comparing job 1 with job 2, job 2 is changed from job 1 by assuming the second machine is broken, the only difference between them is the second machine is broken. But the best process plans of them are completely different. This reveals that an accident in shop floor can lead to the best process plan changed completely. So, it becomes a

Table 5 Best process plan of each job

Job	Best process plan	Production time
1	(1, 2)-(2, 7)-(3, 2)-(4, 2)-(9, 3)	213
2	(10, 9)-(11, 3)-(15, 7)	224

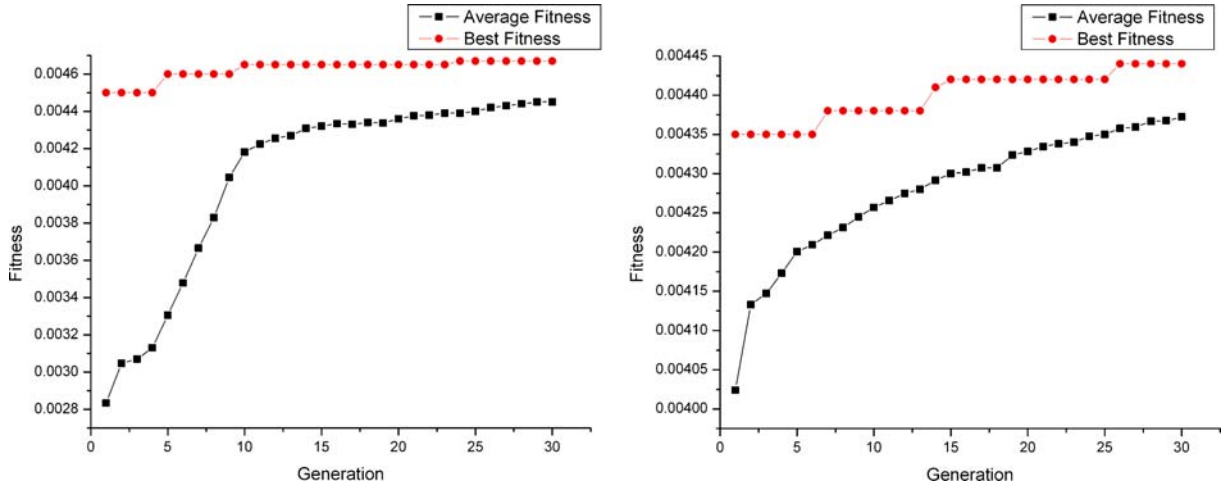

Fig. 8 Convergence curves of GP

very important problem that how to optimize flexible process planning to response to the current shop status quickly. The method which is proposed in this paper used genetic programming to optimize flexible process planning.

The experimental results of Table 4 and Fig. 8 show that GP-based approach can reach good solutions in short time. So, the GP-based approach is a promising method in solving the optimization of flexible process planning problem. And the results also show that the proposed method can reach near-optimal solutions in the early stage of evolution. In order to response to the current shop status, the proposed method can select near optimal process plans quickly and effectively.

5.2 Comparison with GA

The algorithm has been compared with GA. The objective of the experiments is minimizing the production time. The GA is coded in C++, and implemented on the same PC with GP. The GA parameters for the two jobs are given in Table 3, and fitness-proportion selection scheme, single point crossover and point mutation have been used as the reproduction, crossover and mutation operators respectively.

Figure 9 illustrates convergence curves of the two algorithms for two jobs. From the results of Fig. 9, it can be observed that the two approaches can achieve good results. The GP-based approach usually takes less time (less than 30 generations) to find optimal solutions, and the GA -based approach is slower (nearly reach 60 generations) in finding optimal solutions. The GP-based approach also can find near-optimal solutions quicker than the GA- based approach. So, when it is applied to large-scale problems in real world, it is more suitable to reduce much computation time with a little detriment of the solution quality.

Overall, the experiment results indicate that the GP-based approach is a more acceptable optimization approach of flexible process planning.

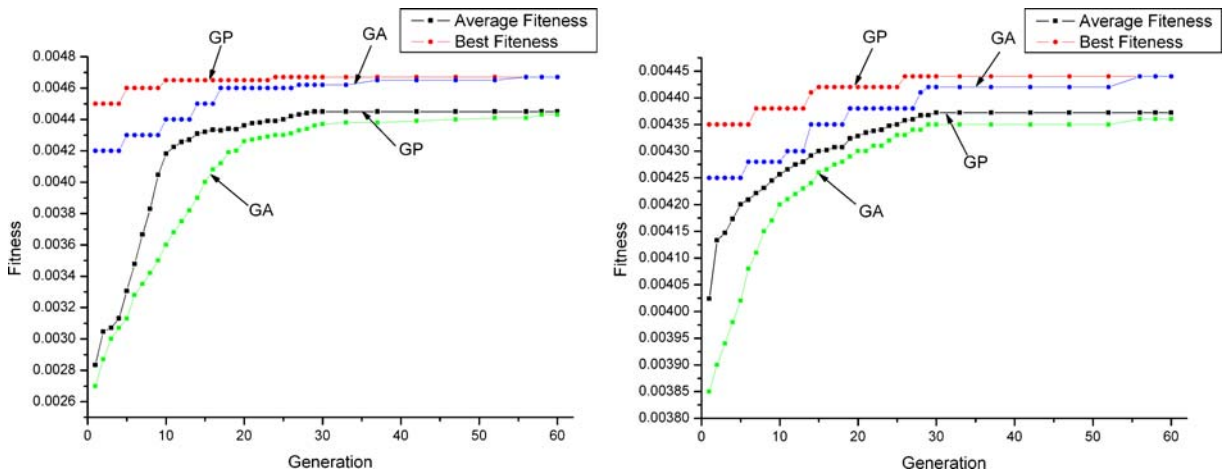

Fig. 9 The comparison between the convergence curves of GP and GA

6 Conclusion

A new approach using genetic programming (GP) is proposed to optimize flexible process planning. The flexible process plans and the mathematical model of flexible process planning have been described, and a network representation is adopted to describe the flexibility of process plans. To satisfy GP, the network has been converted to a tree. The efficient genetic representations and operator schemes also have been considered. Case studies have been used to test the algorithm, and the comparison has been made for this approach and GA, which is another popular evolutionary approach to indicate the adaptability and superiority of the GP-based approach. The experimental results show that the proposed method is a promising and very effective method in the optimization research of flexible process planning.

Although the proposed algorithm in this paper can get good results, testing other genetic operators (reproduction, crossover and mutation) to enhance the efficiency of algorithm is an important future work. With GP-based optimization approach developed in this work, it would be possible to increase the efficiency of manufacturing system. So, one future work is to use the proposed approach to the practical manufacturing system. The increased use of this approach will probably enhance the performances of future process planning systems. Another future research direction is to apply the proposed approach to the integrated environment, such as the integration with scheduling system.

Acknowledgement This research work is supported by 973 National Basic Research Program of China under Grant No.2004CB719405 and 863 High Technology Plan Foundation of China under Grant No.2006AA04Z131.

References

1. Sormaz D, Khoshnevis B (2003) Generation of alternative process plans in integrated manufacturing systems. J Intell Manuf 14:509–526
2. Kusiak A, Finke G (1988) Selection of process plans in automated manufacturing systems. IEEE J Robot Autom 4(4):397–402
3. Bhaskaran K (1990) Process plan selection. Int J Prod Res 28 (8):1527–1539
4. Lee KH, Junq MY (1994) Petri net application in flexible process planning. Comput Ind Eng 27:505–508
5. Kamal R, Jagath S (2003) Processing plan selection algorithms for a cooperative intelligent image analysis system. In: Proceedings of the International Conference on Imaging Science, Systems and Technology, pp 576–582
6. Seo Y, Egbelu PJ (1996) Process plan selection based on product mix and production volume. Int J Prod Res 34(9):2369–2655
7. Usher JM, Bowden RO (1996) Application of genetic algorithms to operation sequencing for use in computer-aided process planning. Comput Ind Eng 30(4):999–1013
8. Tiwari MK, Tiwari SK, Roy D, Vidyarthi NK, Kameshewaran S (1999) A genetic algorithm based approach to solve process plan selection problems. In: IEEE Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials, 1:281–284
9. Rocha J, Ramos C, Vale Z (1999) Process planning using a genetic algorithm approach. In: IEEE Proceeding of International Symposium on Assembly and Task Planning, pp 338–343
10. Dereli T, Filiz HI (1999) Optimisation of process planning functions by genetic algorithms. Comput Ind Eng 36:281–308
11. Moriarty DE, Miikkulainen R (1997) Forming neural networks through efficient and adaptive coevolution. Evol Comput 5:372–399
12. Kramer MD, Zhang D (2000) GAPS: a Genetic Programming System. In: Proceedings of the 24th Annual International Computer Software and Application Conference (IEEE COMPSAC), pp 614–619
13. Banzhaf W, Nordin P (1998) Genetic programming: an introduction. Morgan Kaufmann Publishers, Inc., San Francisco CA
14. Dimopoulos C, Zalzala AMS (2001) Investigating the use of genetic programming for a classic one-machine scheduling problem. Adv Eng Softw 32:489–498
15. Garces PJ, Schoenefeld DA, Wainwright RL (1996) Solving facility layout problems using genetic programming. In: Proceedings of the 1st Annual Conference on Genetic Programming 11 (4):182–190
16. McKay BM, Willis MJ, Hiden HG, Montague GA, Barton GW (1996) Identification of industrial processes using genetic programming. In: Proceeding of the Conference on Identification in Engineering Systems, 1996 11 (5):510–519
17. Hutchinson GK, Flughoeft KAP (1994) Flexible process plans: their value in flexible automation systems. Int J Prod Res 32 (3):707–719
18. Saygin C, Kilic SE (1999) Integrating flexible process plans with scheduling in flexible manufacturing systems. Int J Adv Manuf Technol 15:268–280
19. Benjaafar S, Ramakrishnan R (1996) Modeling, Measurement and evaluation of sequencing flexibility in manufacturing systems. Int J Prod Res 34:1195–1220
20. Kim YK, Park K, Ko J (2003) A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. Comput Oper Res 30:1151–1171
21. Lin YJ, Solberg JJ (1991) Effectiveness of flexible routing control. Int J Flex Manuf Syst 3:189–211
22. Catron AB, Ray SR (1991) ALPS-A Language for Process Specification. Int J Comput Integr Manuf 4:105–113
23. Ho YC, Moodie CL (1996) Solving cell formation problems in a manufacturing environment with flexible processing and routing capabilities. Int J Prod Res 34:2901–2923
24. Koza JR (1990) Genetic programming: a paradigm for genetically breeding populations of computer programs to solve problems. Tech. Rep. STAN-CS-90-1314 Stanford University Computer Science Department
25. Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection and genetics. MIT Press, Cambridge MA
26. Langdon WB, Qureshi A (1995) Genetic Programming - Computers using “Natural Selection” to generate programs. Tech. Rep. RN/95/76, Gower Street, London WCIE 6BT, UK