An Effective Hybrid Genetic Algorithm and Variable Neighborhood Search for Integrated Process Planning and Scheduling in a Packaging Machine Workshop

Li, X., Gao, L., Pan, Q., Wan, L. & Chao, K-M.

Original citation & hyperlink:
Li, X, Gao, L, Pan, Q, Wan, L & Chao, K-M 2018, 'An Effective Hybrid Genetic Algorithm and Variable Neighborhood Search for Integrated Process Planning and Scheduling in a Packaging Machine Workshop' IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. (In-press), pp. (In-press).
https://dx.doi.org/10.1109/TSMC.2018.2881686
DOI 10.1109/TSMC.2018.2881686
ISSN 2168-2216
ESSN 2168-2232

Publisher: IEEE

An Effective Hybrid Genetic Algorithm and Variable Neighborhood Search for Integrated Process Planning and Scheduling in a Packaging Machine Workshop

Xinyu Li, Liang Gao *Member, IEEE*, Quanke Pan, Liang Wan, Kuoming Chao

***Abstract*—Process planning and scheduling were modelled sequentially in traditional manufacturing system. However, because of their complementarity, there is an increasing need to integrate them to greatly enhance the manufacturing productivity. Therefore, the integrated process planning and scheduling (IPPS) becoming a hotspot on providing a blueprint of efficient manufacturing system has attracted more and more attentions. This paper proposes a novel algorithm hybridizing the genetic algorithm with strong global searching ability and variable neighborhood search with strong local searching ability for the IPPS problem. To improve the searching ability, a novel procedure, encoding method and local search method have been designed. Effective operators have been adopted. Three experiments with totally 37 well-known benchmark problems are employed to evaluate the performance of proposed method. Based on the results, the proposed algorithm outperforms the state-of-the-art methods and finds the new solutions (the best solutions found so far) for some problems. The proposed method has also been applied on a real-world case coming from a non-standard equipment production workshop for packaging machine of a machine tool company in China. The solution also shows that it can solve the real-world case very well.**

***Index Terms*—Integrated process planning and scheduling, hybrid algorithm, variable neighborhood search**

I. Introduction

Compared to traditional manufacturing systems, modern manufacturing system consists all of an enterprise's production activities. It is a highly flexible automation production system and has much greater range than ever before. In modern manufacturing systems, process planning and scheduling are two most important subsystems [1].

Process planning which determines raw materials and processes for each job is the link of product design and manufacturing. A process plan includes the machines, fixtures, tools and operations sequence. Because more and more flexible manufacturing systems and CNC machines are used in the modern manufacturing system, a job may have several alternative process plans [2]. The Computer Aided Process Planning (CAPP) system can convert the product design information from Computer Aided Design (CAD) to manufacturing information [3].

The scheduling system is to arrange all the operations on relative machines to satisfy the process plans constraints and optimize the predefined objectives [4-9]. Scheduling is the link of two production steps which are the preparing processes and putting them into action [10-12].

Therefore, the relationship between process planning and scheduling is very close. However, in traditional approaches, process planning and scheduling were carried out independently, where the scheduling was optimized after all the process plans had been already generated [13-14]. Most researchers do not focus on the integration of them in past. Although optimizing the process plans can improve the usage of resources and scheduling can optimize the production efficiency separately, the traditional method still impedes the further improvement of the productivity and responsiveness of the manufacturing systems. It also would bring some problems, including the lack of flexibility in process plan, conflicting optimization objectives between them, unbalanced machines load and so on [15]. Because of the development of the modern manufacturing system (using more and more flexible manufacturing systems and CNC machines), the CAPP system can produce several alternative process plans for every job. So, there is an increasing need to integrate the process planning and scheduling more closely. It can significantly improve the manufacturing efficiency through eliminating scheduling conflicts, improving resources utilization, adapting to irregular workshop floor disturbances and so on.

However, the IPPS problem is very different from the separated process planning and scheduling. The IPPS problem which is NP-hard contains more constraints. The previous methods of scheduling cannot be applied on IPPS problem directly. For searching an optimized solution efficiently in complex solution space, it is necessary that the method can balance its diversification and intensification searching abilities.

Manuscript received ****. This work was supported in part by the National Natural Science Foundation of China under Grant 51825502, 51775216, 51435009 and 51711530038; in part by the Natural Science Foundation of Hubei Province under Grant 2018CFA078; and in part by the Program for HUST Academic Frontier Youth Team. *(Corresponding author: Liang Gao.)*

X. Y. Li, L. Gao, Q. K. Pan and L. Wan are with the State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: lixinyu@hust.edu.cn; gaoliang@mail.hust.edu.cn; panquanke@hust.edu.cn; 371116194@qq.com).

K. M. Chao is with the Department of Computer and Network Systems, Coventry University, Coventry CV1 5FB, UK (e-mail: csx240@coventry.ac.uk).

However, a single algorithm has its own limitations and is difficult to satisfy the above two requirements simultaneously. For example, genetic algorithm (GA) has strong global searching ability but lacks local searching ability, and variable neighborhood search (VNS) has strong local searching ability but lacks global searching ability [16]. Therefore, a hybrid algorithm is a good way to solve the IPPS problem, which has become a new trend [14] [17] [18]. In this paper, a new hybrid genetic algorithm and variable neighborhood search (GAVNS) has been proposed to optimize the IPPS problem. The experimental studies can show the advantages of this hybrid algorithm clearly. This method has also been applied to solve a real-world complex manufacturing case successfully.

The remainder of the paper is organized as follows. Section II reviews the related literature. Section III presents the problem formulation. Section IV describes the proposed GAVNS for integrated process planning and scheduling. Section V shows some experimental studies. Section VI presents a real-world case study. Section VI concludes the paper and discusses our future work.

II. Literature Survey

IPPS is an important and momentous issue in improving manufacturing productivity, some studies of IPPS have been reported. Chryssolouris *et al.* [19] firstly proposed the concept of IPPS. Tan *et al.* [20] presented the first review of earlier IPPS researches. Kim *et al.* [21] presented the most well-known benchmark problems for IPPS.

Currently, several IPPS integration models have been established and can be divided into 3 categories: nonlinear process planning (NLPP), closed loop process planning (CLPP) and distributed process planning (DPP) [22-23]. NLPP is a basic IPPS model. As NLPP model has a simple integration method and easy to operate, the most current researches of IPPS focus on this model. Based on a static manufacturing environment, the NLPP generates many alternative plans for every job, and every alternative process plan is assigned a priority based on the objective of process planning. The plan with highest priority is first provided to the scheduling system. If the highest priority plan is not suitable for the current workshop status, the sub-optimal priority plan will be provided.

The main methods of IPPS include artificial intelligence (AI) based approaches, local search (LS) approaches, etc.

A. AI based Approaches

The main approaches in this category include evolutionary algorithm (EA), multi-agent system (MAS) approach, etc.

Li *et al.* [24] proposed a EA based method. Zhang *et al.* [25] proposed an object-coding GA to solve the IPPS. Luo *et al.* [26] presented a multi-objective GA for the multi-objective IPPS. Shokouhi [27] developed a GA based method to consider all the alternative process plans. Qiao *et al.* [15] proposed an improved GA for IPPS.

Some researchers used the MAS approach to study the IPPS problem [28-29]. Li *et al.* [22] presented an agent-based method. Wong *et al.* [30] implemented a MAS with a two-stage colony optimization algorithm (ACO). Mishra *et al.* [31] developed a self-reactive cloud-based multi-agent architecture.

What's more, there are several other AI approaches, including particle swarm optimization (PSO) algorithm [32-33], ACO [3][34][35], and so on.

B. Local Search Approaches

Local search is anthor important approach for the scheduling problems and several researchers also use it for the IPPS problem. Li *et al.* [36] developed a simulated annealing (SA) to optimize the IPPS problem. Chan *et al.* [37] proposed an enhanced swift converging SA algorithm. Li *et al.* [38] proposed a hybrid algorithm combing the tabu search (TS) as the local search method. Sobeyko *et al.* [16] proposed a VNS to solve the IPPS problem in large-scale flexible job shops. Naseri *et al.* [39] presented a hybrid GA which incorporating a local search algorithm for this problem. Because of the difficulty in neighborhoods design, the reported local search algorithms for the IPPS problem are few. However, also because of the effective neighborhood approach, it can solve the IPPS problems effectively. So using local search algorithms to solve the IPPS problem has broad prospect.

In addition to the above approaches, there are several other approaches to solve the IPPS problem, such as mathematical programming, constraint satisfaction, heuristic rules and coevolution [23]. Each approach has its own advantages and disadvantages. How to take their advantages, so that they can deal with the IPPS problem better, is the current research focus.

A new hybrid algorithm presented in this paper hybridizing the GA with strong global searching ability and VNS with strong local searching ability is proposed to optimize the IPPS.

III. Problem Formulation

A. Problem Definition

The description of IPPS problem is as follow [24]: Given a set of n jobs with alternative process plans and m machines in the workshop, optimize the schedule and process plans for all the jobs simultaneously according to constraints and objectives.

There are three major kinds of process plan flexibility: machine flexibility, sequencing flexibility and processing flexibility. Machine flexibility refers that a process can be operated by different machines. Sequencing flexibility refers that the sequence of some machining features can be exchanged. Because, not all of the features have strict sequence constraints, so there are a variety of operations sequences of the same job. Processing flexibility refers that the same feature can be operated through different processes, which is resulted from the various processing schemes of each manufacturing feature and each scheme has different processing methods.

A network for flexible process plans proposed by Kim *et al.* [21] is adopted here. Fig. 1 is a flexible process plans network for a job. This network is a single-direction acyclic graph consisting three kinds of nodes and the directed edge of the precedence relationship between describing nodes. In Fig. 1, S and E nodes are virtual nodes, and S node represents the starting node and E node represents the ending node. Boxes represent the processing operation nodes. Arrows indicate the

processing sequence. OR represents the processing flexibility that is to say this feature can be processed by different processes. If a subsequent route of one node is connected by an OR, then a job can be processed only through one OR. OR route indicates that a processing path, which is from OR to JOIN. If the path does not connect with OR, all the operations on this line must be visited. For example, a feasible route of the job is: $O_1(M_{11})$-$O_2(M_{12})$-$O_6(M_5)$-$O_7(M_9)$-$O_{12}(M_7)$-$O_8(M_{12})$-$O_9(M_2)$-$O_{13}(M_1)$-$O_{11}(M_8)$-$O_{14}(M_3)$-$O_{15}(M_1)$. The first number represents the corresponding operation; the second one represents the selected machine for this operation.

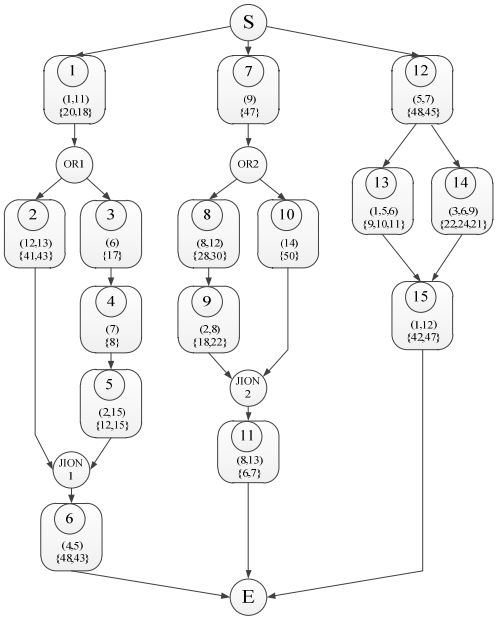

Fig. 1. Flexible process plans network for a job

B. Mathematical Modeling of IPPS

It is generally assumed in traditional scheduling: there is only one process plan for each job, which indicates that process planning does not take the flexibility into account. Because of the increasing use of flexible manufacturing systems and CNC machines, each job can have more than one feasible process plans. Moreover, the scheduling determines the optimal process route for each job and completes all the jobs based on the specific objective without violating the constraints of manufacturing environment. The makespan is used here to optimize the IPPS problem. Its mathematical model can refer to our previous work [24].

IV. Proposed Hybrid GAVNS for IPPS

The IPPS problem is NP-hard. The traditional solution usually uses only one algorithm. However, studies show that every algorithm has its own advantages and disadvantages. It's difficult to solve the complicated IPPS problem using only one algorithm, while hybrid algorithm maybe better. GA is a global optimization algorithm simulating the mechanism of natural evolution. It is of implicit parallelism and global search performance in searching solution space. It provides a method which can be easily hybridized with other local search algorithm. VNS algorithm is a fast and efficient local search algorithm in solving complex combination optimization problems. However, the traditional GA has the inefficient local search ability, and the performance of VNS strongly depends on the initial solutions and neighborhood structures. An efficient hybrid algorithm GAVNS can be designed to embed the VNS into the GA. GA provides better initial solutions for the VNS to exploit them, so the solution space can be effectively explored and exploited, as VNS provides concentric neighborhood search for the local area. Thus, the search ability of global search and local search can both be improved in solving the IPPS problems.

A. The Procedure of Proposed GAVNS Algorithm

According to the characteristic of IPPS problem, the GAVNS is designed. The part of process planning uses GA to optimize the process plan for each job. To make the access to process planning section and get near-optimal process plan easier, the genetic operation is repeated for i times (the range of i is suggested from 1 to 20) before generating new population, in which the larger the problem size is and the bigger the i value is, the easier to get the optimized solutions. When the number of jobs is less than 10 for the problem, the i can be set as 1-6. For the large scales of problems (the number of jobs is bigger than 15), the i can be set as 10-20. The part of scheduling based on GAVNS outputs the optimal schedule and process plan of every job with the best objective. The merit of this optimization procedure is considering both the process flexibility and computation time.

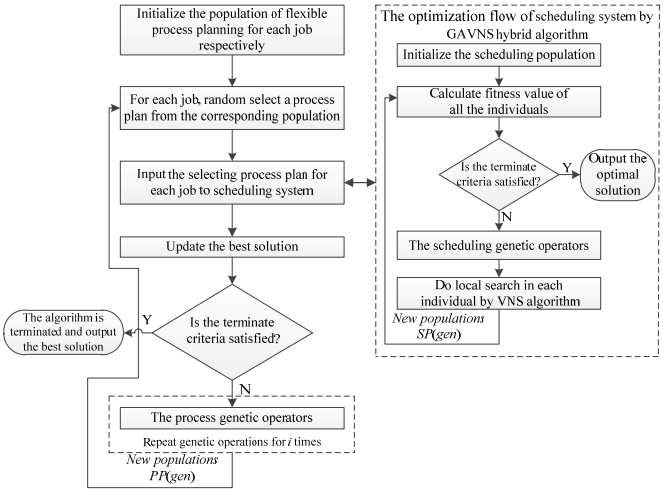

Fig. 2. Workflow of the proposed GAVNS

The workflow of proposed GAVNS is shown in Fig. 2:

Step 1: Respectively initialize the process planning population (PPn (0)) of every job.

Step 2: Select a process plan randomly for every job and put it into the scheduling system.

Step 3: Use the GAVNS to optimize the scheduling.

Step 3.1: According to the selected process plan of every job, generate the initial scheduling population.

Step 3.2: Evaluate the scheduling population (SP(0)).

Step 3.3: If the terminate criteria of scheduling is satisfied, output the optimal solution. Otherwise, go to Step 3.4.

Step 3.4: Generate the new scheduling individuals by genetic

operators.

Step 3.5: Use the VNS algorithm to update every individual.

Step 3.6: Generate the new scheduling population *SP*(*gen*), go to Step 2, *gen* = *gen*+1.

Step 4: Update the best solution.

Step 5: If the terminate criteria of process planning system is satisfied, output the optimal solution. Otherwise go to Step 6.

Step 6: Generate the new process planning population *PP*(*gen*).

Step 7: Go to Step 2, *gen* = *gen*+1.

B. Genetic Operations of Process Planning

1) Encoding and Decoding

The encoding and decoding methods in Li *et al.* [2] are employed for the process planning population.

TABLE I
FLEXIBLE PROCESS PLANS INFORMATION

Features	Alternative operations	Alternative machines	Processing time	Precedence constraints
F_1	O_1	M_1, M_{11}	20, 18	Before F_2, F_3
F_2	O_2	M_{12}, M_{13}	41, 43	Before F_3
	O_3-O_4-O_5	M_6/ M_7/ M_2, M_{15}	17/ 8/ 12, 15	
F_3	O_6	M_4, M_5	48, 43	
F_4	O_7	M_9	47	Before F_5, F_6
F_5	O_8-O_9	M_8, M_{12}/ M_2, M_8	28, 30/ 18, 22	Before F_6
	O_{10}	M_{14}	50	
F_6	O_{11}	M_8, M_{13}	6, 7	
F_7	O_{12}	M_5, M_7	48, 45	Before F_8, F_9, F_{10}
F_8	O_{13}	M_1, M_5, M_6	9, 10, 11	Before F_{10}
F_9	O_{14}	M_3, M_6, M_9	22, 24, 21	Before F_{10}
F_{10}	O_{15}	M_1, M_{12}	42, 47	

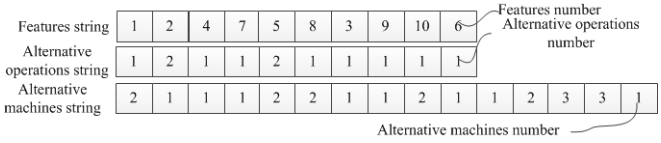

Fig. 3. Multi-part encoding method for process planning

The multi-part encoding method consists of three parts, namely, feature string, alternative operations string and alternative machines string [2]. The process plans of the job in Fig. 1 are shown in Table I. Fig. 3 gives an example of the encoding method. The job contains 10 features with 15 operations. So, the feature string and alternative operations string are made up of 10 elements, and the alternative machine string is made up of 15 elements. The feature string is a permutation from 1 to 10 under the order constraint satisfaction of features, of which the processing sequence is F_1-F_2-F_4-F_7-F_5-F_8-F_3-F_9-F_{10}-F_6. For the alternative operations string, the second element is 2. It means that the 2nd feature (F_2) of the job selects its 2nd alternative process, i.e., O_3-O_4-O_5. The number on the 5th gene position is 2, indicating that F_5 selects the 2nd operation (O_{10}) from its alternative operation set. For the alternative machine string, the first element is 2. It means that the 1st operation (O_1) selects its 2nd alternative machine, i.e., M_{11}. The number on the 15th gene position is 1, indicating that O_{15} selects the 1st machine from its alternative machines set, i.e., M_1.

Decoding method is to translate the code into a solution of the problem, is as follow:

Step 1: First, decode an alternative operations string, then get the alternative operation selected by each feature from left to right in turn. As shown in Table I and Fig. 3, the obtained alternative operations selected by various features are: F_1 (O_1), F_2 (O_3-O_4-O_5), F_3 (O_6), F_4 (O_7), F_5 (O_{10}), F_6 (O_{11}), F_7 (O_{12}), F_8 (O_{13}), F_9 (O_{14}), and F_{10} (O_{15});

Step 2: Decode the features string, and the string just represents the processing sequence of feature. As shown in Table I and Fig. 3, the obtained processing sequence of each feature sequence is: F_1-F_2-F_4-F_7-F_5-F_8-F_3-F_9-F_{10}-F_6. Then the result obtained from *Step* 1 can bring out the processing sequence of each operation is O_1-O_3-O_4-O_5-O_7-O_{12}-O_{10}-O_{13}-O_6-O_{14}-O_{15}-O_{11};

Step 3: Decode the alternative machines string, then get the processing machine selected for each operation from left to right in turn. As shown in Table I and Fig. 3, the obtained processing machines are: O_1 (M_{11}), O_2 (M_{12}), O_3 (M_6), O_4 (M_7), O_5 (M_{15}), O_6 (M_5), O_7 (M_9), O_8 (M_8), O_9 (M_8), O_{10} (M_{14}), O_{11} (M_8), O_{12} (M_7), O_{13} (M_6), O_{14} (M_9), and O_{15} (M_{12});

Step 4: According to the results from *Step* 2 and *Step* 3, determine the final routing of the job, and the processing machine and processing time corresponding to each operation. As shown in Table I and Fig. 3, the operation route the encoding scheme obtains is: $O_1(M_{11})$-$O_3(M_6)$-$O_4(M_7)$-$O_5(M_{15})$-$O_7(M_9)$-$O_{12}(M_7)$-$O_{10}(M_{14})$-$O_{13}(M_6)$-$O_6(M_5)$-$O_{14}(M_9)$-$O_{15}(M_{12})$-$O_{11}(M_8)$. The processing time corresponding to each operation can be obtained from Table I.

2) Initialization

Based on the encoding method mentioned above, the process planning population is initialized randomly. Randomly generates a feasible sequence from 1 to *n* (*n* represents the total number of the job) that is modeled as feature string; randomly select an operation from the alternative operation set corresponding to every feature as the operation of the feature to be encoded in alternative operation string; randomly select a machine from alternative machine set corresponding to every operation as the machine of the operation to be represented by alternative machine string. To enable the process planning system to select broader process plan, there is no fitness evaluation in process planning.

3) Genetic Operators of Process Planning

Selection: To enable the process planning system to select broader process plan, the selecting operation of process planning uses random selection method.

Crossover: The crossover operators in Li *et al.* [2] are employed for the process planning population.

Crossover operator of feature string is shown in Fig. 4, and the steps are as follows:

Step 1: Two crossover points are randomly selected. And two chromosomes of the parent are divided into three parts: left, middle and right, are represented as A, B and C;

Step 2: Replicate the genes in the middle part of the parent chromosomes into the offspring;

Step 3: Remove the existing gene of offspring 1 from parent 2, and then fill the left genes of the parent 2 in the blank space of offspring 1 in turn, and offspring 2 can be obtained in the same way.

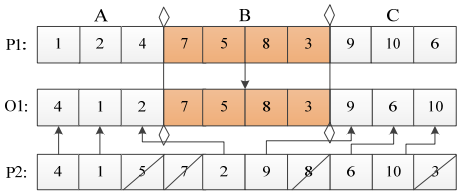

Fig. 4. Crossover operation of feature string

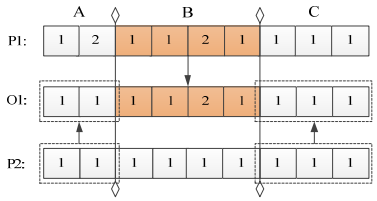

Fig. 5. Crossover operation of alternative operation string

Crossover operator of alternative operation string is shown in Fig. 5, and the steps are as follows:

Step 1: Randomly select two crossover points, the two parent chromosomes are divided into three parts: left, middle and right, are represented as A, B and C;

Step 2: Replicate the genes in the middle part of the parent chromosomes into the offspring;

Step 3: A and C in parent 2 are copied to the positions corresponding to offspring 1, and offspring 2 can also be obtained in the same way.

Crossover operator of alternative machine string is shown in Fig. 6. The process steps are the same as the crossover operator of the alternative operation string.

Mutation: The two-point swapping mutation is adopted for the feature string. For the alternative operations string and alternative machines string, the mutation randomly selects one element in the string, and changes its value to another one in the corresponding range.

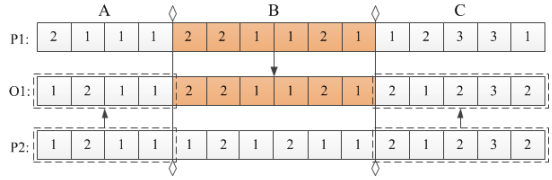

Fig. 6. Crossover operation of alternative operation string

C. Genetic Operations of Scheduling

1) Encoding and Decoding

The operation-based representation is used as the encoding method of scheduling population. This representation contains a permutation with P_i repetitions of job numbers. By scanning the chromosome from left to right, the kth appearance of a job number means the kth operation of the job. The merit of this representation is that any permutation of the chromosome can be decoded to a feasible solution. And, the chromosome is decoded into active schedule to get a better solution.

2) Initialization and Fitness

Based on the encoding method mentioned above, the initial population can be generated randomly. In the scheduling, the makespan is used as the fitness objective.

3) Genetic Operators of Scheduling

Selection: The tournament selection method is adopted here. In this method, two individuals are selected randomly. If a random value (between 0 and 1) is less than the pre-defined probability, choose the individual with better fitness. Otherwise, choose the other one. The reproduction probability is set as 0.8 in this paper.

Crossover: The Precedence Operation Crossover (POX) and Job-based Order Crossover (JOX) are used as the crossover operator for the scheduling population here. During the running of GA, randomly select one as the current crossover operator.

As is shown in Fig. 7, the steps of POX are as follows:

Step 1: Randomly divide the set of job numbers, {1, 2, 3, ..., n}, into two non-empty subsets as *JobSet*1 and *JobSet*2;

Step 2: Duplicate the job numbers from parent 1 which are contained in *JobSet*1 into offspring 1, and duplicate the job numbers from parent 2 which are contained in *JobSet*1 into offspring 2 and keep their positions;

Step 3: Duplicate the job numbers from parent 2 which are contained in *JobSet*2 into offspring 1, and duplicate the job numbers from parent 1 which are contained in *JobSet*2 into offspring 2 and keep their sequence at the same time.

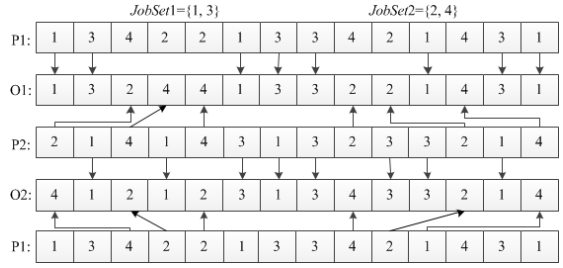

Fig. 7. POX operation for scheduling

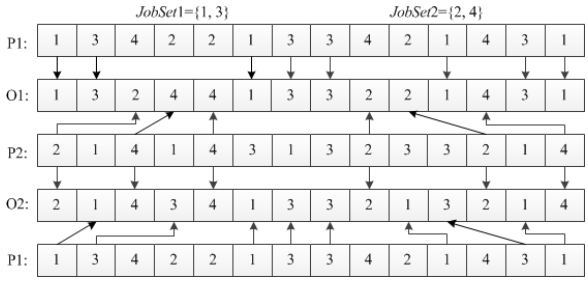

Fig. 8. JOX operation for scheduling

As is shown in Fig. 8, the steps of JOX are as follows:

Step 1: Randomly divide the set of job numbers, {1, 2, 3, ..., n}, into two non-empty subsets as *JobSet*1 and *JobSet*2;

Step 2: Duplicate the job numbers from parent 1 which are contained in *JobSet*1 into offspring 1, and duplicate the job numbers from parent 2 which are contained in *JobSet*2 into offspring 2 and keep their positions at the same time;

Step 3: Duplicate the job numbers from parent 1 which are contained in *JobSet*1 into offspring 1, and duplicate the job numbers from parent 2 which are contained in *JobSet*1 into offspring 2 and keep their sequence at the same time.

Mutation: The two-point swapping mutation and two fragments exchange mutation are used as the mutation operator for the scheduling here. During the running of genetic algorithms, select one as the current mutation operator randomly. The two-point swapping mutation has already had detailed description in the part of process planning. The two fragments exchange mutation is to select two gene segments randomly from the parent chromosomes, and then exchange these two segments to get offspring.

D. Local Search Procedure for Scheduling

1) Basic Principles of VNS

The relationship among local optimal solution, global optimal solution and neighborhood structure in combinatorial optimization problem can be illustrated as follows:

- For different neighborhood structures, the local optimal solution in a neighborhood structure is not necessarily the best local optimal solution in another neighborhood.
- For all the neighborhood structures, the global optimal solution is necessarily the local optimal solution in a certain neighborhood.
- For most combinatorial optimization problems, the local optimal solutions in a neighborhood structure are usually another near the local optimal solutions in another neighborhood.

VNS algorithm is an efficient local search method proposed by Mladenovic *et al.* [40] based on the relationship mentioned above. Beginning with an initial solution and taking advantages of neighborhood structures, the algorithm searches the better one than the current one in the neighborhood of the current solution continuingly. If a new solution is found, the current solution will be replaced. The repetition of iteration ends until a termination condition is satisfied. The basic concept of VNS algorithm is to change multiple neighborhood structures systematically within the local search range. The neighborhood structure of VNS is not unitary and fixed, and it has multiple neighborhood structures. With one same initial solution, it has more extensive and in-depth search space compared with other searches, and it can get rid of local optimal solution easily and the approximate optimal solution at a faster speed. VNS algorithm is not necessarily to adjust the parameters adaptively and it is very effective, so it has been successfully used in the combinatorial optimization problems.

The basic procedure of VNS is shown in Fig. 9 [40]:

Step 1: Define a series of neighborhood structure, N_k, k = 1, 2, …, k_{max}.

Step 2: Set k = 1, and repeat the following procedure until k = k_{max}:

Step 2.1: Local Search, in the current neighborhood N_k, from the current solution S began searching through the local neighborhood to get the best solution S_1.

Step 2.2: Move or Not, if S_1 is better than S, then order $S = S_1$, start the search again from k = 1; otherwise, let k = k+1, continue to search for the next neighborhood.

2) Neighborhood Structures

VNS algorithm mainly uses the concept of systematic transformation of multiple neighborhood structures to enhance the local search ability and avoid falling into the local optimum. Whether the neighborhood structures design is reasonable or not will exert a direct impact on the searching performance of VNS algorithm. For job shop scheduling, the critical path directly affects the makespan of a schedule. The movement of neighborhood is usually conducted through subtle perturbations to the operation on critical path. And only using this could shorten the makespan of finding out the current solution. Combining with the characteristics of VNS algorithm, this research uses three neighborhood structures to deal with the IPPS problem. They are the N5 neighborhood based on the critical block search, total neighborhood based on random search and randomly inserted neighborhood.

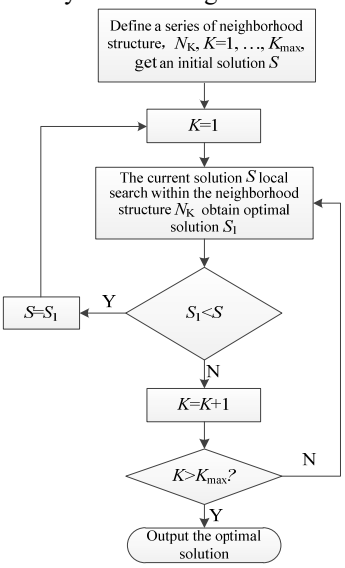

Fig. 9. Workflow of the VNS

N5 neighborhood based on critical block search: N5 neighborhood structure was presented by Nowicki *et al.* [41]. Its principles are that only the two critical operations at block end can be exchanged specifically to the first critical block in critical path. Only the two critical operations at block head can be exchanged specifically to the last critical block in critical path. Only the two adjacent critical operations at block end and block head can be exchanged specifically to the middle block of the critical path, in avoidance of the exchange of the operations within blocks. Any exchange will not be conducted if there is only one operation in critical blocks. To avoid generating non-feasible solution, the two operations of the same job shall not be exchanged. Moreover, the critical block mentioned above refers to the set of the operations on the critical path continuingly conducted on the same machine.

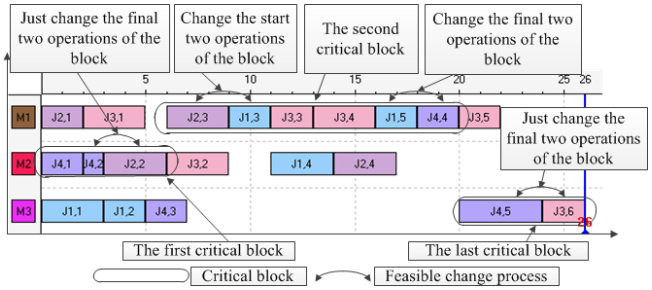

Fig. 10. N5 neighborhood structure

Fig. 10 shows the diagram of the critical blocks adjustment of N5 neighborhood structure. It represents the Gantt chart of a feasible job scheduling which is a scheduling scheme of 20 operations generated by four jobs and three machines. For this scheduling problem, the movements of N5 neighborhood structure includes: {(O_{42}→O_{22}), (O_{23}→O_{13}), (O_{15}→O_{44}), (O_{45} →O_{36})}.

Total Neighborhood based on Random Search: This neighborhood design stems from the mutation operation. It evaluates the neighborhood formed by the full permutations of λ different genes on the chromosome. The specific steps are: first, randomly to select λ genes from parent chromosomes, of which the genes' value cannot be the same and the genes can generate all neighborhoods of their permutations; second, to select the best individual as offspring chromosomes to evaluate all the chromosomes in neighborhoods.

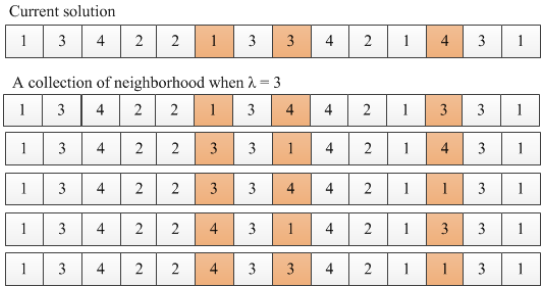

Fig. 11. Total neighborhood based on random search

As it can be seen in Fig. 11, a total neighborhood search process of a chromosome consists of four jobs. It randomly selects certain chromosomes located on the 6th, 8th and 12th positions from parent chromosomes. They are respectively "1", "3" and "4", and then generate all the neighborhood chromosome of their permutation to evaluate all these chromosomes to select the optimal individual as offspring chromosomes.

Randomly Inserted Neighborhood: It belongs to the simple inserted neighborhood. That is to say, to select one or several operations and randomly insert them into certain position located in coding sequence, which forms a new coding sequence.

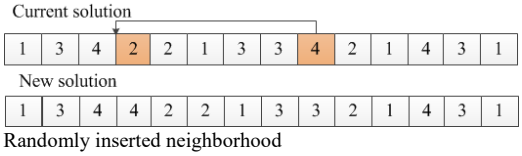

Fig. 12. Randomly inserted neighborhood

As it can be seen in Fig. 12, a randomly inserted neighborhood search process of a chromosome consists of four jobs. It randomly selects the genes in the 4th and 9th position from parent chromosomes, and they are respectively "2" and "4". And then neighborhood chromosome will be obtained if the Gene"4" in 9th position is inserted ahead of 4th position. After acquirement of a specified number of neighborhood chromosomes by the way mentioned above, the optimal individual will be the offspring chromosomes after the evaluation of these neighborhood chromosomes.

V. Experimental Studies and Discussions

In this paper, to evaluate the performance of the proposed GAVNS, 3 Experiments which contains totally 37 problems are adopted from other literature. All of them are the well-known benchmark problems in the IPPS area. The results of the proposed GAVNS are compared with the results of the state-of-the-art algorithms. The parameters of the proposed GAVNS are given in Table II.

The proposed GAVNS algorithm is coded in C++ and implemented on a computer with a 2.8 GHz Core (TM) i7-7600u CPU and 16GB RAM. For every problem, the algorithm runs 10 independent times. The CPU time of proposed GAVNS algorithm is the average of 10 independent running times.

TABLE II
The GAVNS Parameters

Parameters (process planning)	Value	Parameters (scheduling)	Value
PPopSize	100	*SPopSize*	200
PMaxGen	50	*SMaxGen*	100
PP_r	0.02	SP_r	0.02
PP_c	0.80	SP_c	0.70
PP_m	0.10	SP_m	0.20

A. Experiment 1

Experiment 1 includes 8 problems adopted from Moon *et al.* [42] and Naseri *et al.* [39]. P1-1, P1-2 and P1-3 are the extensions of P1. P2-1, P2-2 and P2-3 are the extensions of P2. Table III shows the experimental results and the comparisons with the state-of-the-art methods. The experimental results indicate that the proposed GAVNS can get the better results with less CPU time than the reported methods. And the average results of GAVNS algorithm are better than other algorithms for P2-2 and P2-3.

B. Experiment 2

Experiment 2 includes 5 problems named as P3-P7 adopted from Chan *et al.* [43], Zhang *et al.* [34], Li *et al.* [24], Shao *et al.* [13] and Leung *et al.* [44] respectively. Table IV shows the experimental results and the comparisons with the state-of-the-art methods. The experimental results also indicate that the proposed GAVNS can get the better solutions with less CPU time than the reported methods. Except P4, for other 4 benchmark problems, the proposed algorithm has found the new solutions (the best solutions found so far). And the CPU time of proposed algorithm are much less than other algorithms.

C. Experiment 3

Experiment 3 including 24 problems is adopted from Kim *et al.* [21] [47], which is the most well-known benchmark for IPPS. Many researchers had used it to evaluate the performances of their proposed methods. The 24 problems are constructed with 18 jobs and 15 machines.

Table V shows the experimental results and the comparisons with the state-of-the-art methods including symbiotic evolutionary algorithm (SEA) [21], imperialist competitive algorithm (ICA) [48], improved genetic algorithm (IGA) [15], active learning genetic algorithm (ALGA) [1], object-coding genetic algorithm (OCGA) [25], ant colony optimization (ACO) [34], cross-entropy-based approach (CE) [45] and enhanced ant colony optimization (EACO) [3]. The results of Experiment 3 can also indicate that the proposed GAVNS can get the better solutions with less CPU time than other algorithms.

From the Table V, 19 results of the proposed GAVNS algorithm are either same or better than the other methods. For 6 benchmark problems (problems 17, 18, 20, 22, 23 and 24), the proposed method has found new solutions (the best solutions found so far). The advantage of proposed GAVNS algorithm for the large scale problems (problem 16-24) can be shown more clearly. In addition, 13 solutions of GAVNS reach to the low bound. The low bound is the possible optimal result and the actual optimal solution is not to be better than it [15]. If the actual solution is equal to the low bound, this solution will be the optimal result of this problem. The low bounds in Table V are adopted from Qiao *et al.* [15].

The average of the makespan for the total 24 problems also can show that the proposed method has achieved good improvement. And, for the CPU time, the proposed algorithm also costs much less time than the other algorithms. Based on the average of CPU time for the total 24 problems, the proposed method can save much computation time. The Gantt chart of Problem 24 is shown in Fig. 13.

The results of Experiments 1-3 show that GAVNS can more easily obtain the best solutions of IPPS problems with less CPU time. Its merit can be shown clearly.

TABLE III
THE EXPERIMENTAL RESULTS OF EXPERIMENT 1

Problem	No. of Jobs	No. of Machines	Best		Average		CPU Time (s)		Adopted from
			Reported	**GAVNS**	Reported	**GAVNS**	Reported	**GAVNS**	
P1	5	5	**16**	**14**	/	**14**	/	**1.84**	Moon *et al.* [42]
P1-1	25	5	**62**	**62**	**62**	**62**	5	**3.20**	Naseri *et al.* [39]
P1-2	50	5	**124**	**124**	**124**	**124**	32	**15.24**	
P1-3	100	5	**247**	**247**	247.3	**247**	267	**85.15**	
P2	8	6	**23**	**23**	/	**23**	/	**1.85**	
P2-1	40	6	**105**	**105**	**105**	**105**	27	**12.63**	
P2-2	80	6	**208**	**208**	208.1	**208**	196	**85.28**	
P2-3	160	6	**416**	**416**	416.4	**416**	635	**508.15**	

TABLE IV
THE EXPERIMENTAL RESULTS OF EXPERIMENT 2

Problem	No. of Jobs	No. of Machines	Best Result				Average Result	CPU Time (s)	
P3	8	5	SA[43]	Hybrid TSA[43]	AIS-FLC[43]	**GAVNS**	**GAVNS**	/	**GAVNS**
			30	28	26	**24**	**24**	/	**1.57**
P4	2	4	ACO [34]			**GAVNS**	**GAVNS**	ACO[34]	**GAVNS**
			59			**59**	**59**	1.8	**0.82**
P5	6	5	No Integration[24]	EA[24]	CE[45]	**GAVNS**	**GAVNS**	CE[45]	**GAVNS**
			102	92	91	**90**	**90**	8.4	**5.32**
P6	6	8	HIA [13]	MGA [13]	CE[45]	**GAVNS**	**GAVNS**	CE[45]	**GAVNS**
			250	162	155	**128**	**128**	10.6	**5.85**
P5	5	3	ACO_Agent10[44]	HA [38]	IGA[15]	PGA[46]	**GAVNS**	**GAVNS**	/	**GAVNS**
			380	360	360	360	**350**	**350**	/	**5.13**

TABLE V
THE EXPERIMENTAL RESULTS OF EXPERIMENT 3

Problem	No. of jobs	Makespan										CPU time(s)					
		SEA [21]	ICA [48]	IGA [15]	ALGA [1]	OCGA [25]	ACO [34]	CE [45]	EACO [3]	**GAVNS**	Low bound	IGA [15]	OCGA [25]	ACO [34]	EACO [3]	**GAVNS**
1	6	428	427	427	427	427	427	427	427	**427**	427	11	4.5	4.1	17	**2.18**
2	6	343	343	343	343	343	343	343	343	**343**	343	11	6.5	4	15	**2.38**
3	6	347	345	344	344	344	344	344	344	**344**	344	11	7.9	4.9	14	**2.2**
4	6	306	306	306	306	306	307	306	306	**306**	306	8	4.4	3	14	**1.85**
5	6	319	319	304	321	318	318	315	318	318	304	8	6	3	11	**1.83**
6	6	438	435	427	427	427	427	429	427	**427**	427	13	7.4	7	20	**2.27**
7	6	372	372	372	372	372	372	372	372	**372**	372	9	4.1	3	11	**1.47**
8	6	343	343	342	347	343	343	343	343	343	342	17	6.2	4	13	**2.12**

9	6	428	427	427	427	427	427	427	427	**427**	427	9	5.7	6	21	**2.23**
10	9	443	440	427	427	427	427	427	427	**427**	427	17	10.9	10	34	**5.33**
11	9	369	367	368	369	348	364	365	348	349	344	16	12.2	9.4	31	**5.41**
12	9	328	327	312	327	318	332	322	322	319	306	13	8.7	6.8	24	**4.3**
13	9	452	457	429	436	427	427	433	427	**427**	427	19	15.3	13	39	**7.41**
14	9	381	390	386	380	372	382	398	373	379	372	16	11.2	8	26	**5.86**
15	9	434	432	427	427	427	427	427	427	**427**	427	14	10.7	12	33	**8.4**
16	12	454	466	433	446	427	438	448	429	**427**	427	23	27.8	16.5	50	**10.78**
17	12	431	443	415	423	370	398	424	377	**362**	344	23	27.5	18.7	64	**12.76**
18	12	379	384	364	377	351	378	375	357	**349**	306	20	26.4	15.3	53	**12.07**
19	12	490	490	450	474	427	451	480	431	**427**	427	28	30.5	21	78	**15.56**
20	12	447	440	429	438	384	412	430	386	**383**	372	26	25.9	15.1	55	**12.91**
21	12	477	466	433	447	427	430	442	428	**427**	427	24	26.5	20.1	67	**15.38**
22	15	534	529	491	513	446	480	512	444	**433**	427	27	33.5	30.1	121	**19.43**
23	15	498	495	465	470	394	453	471	413	**388**	372	26	31.5	26	93	**19.08**
24	18	587	577	532	548	458	525	528	460	**446**	427	39	48.5	40	186	**27.49**
Average		417.8	417.5	402.2	409	387.9	401.3	407.8	389.8	**386.5**		17.8	16.7	12.5	45.4	**8.4**

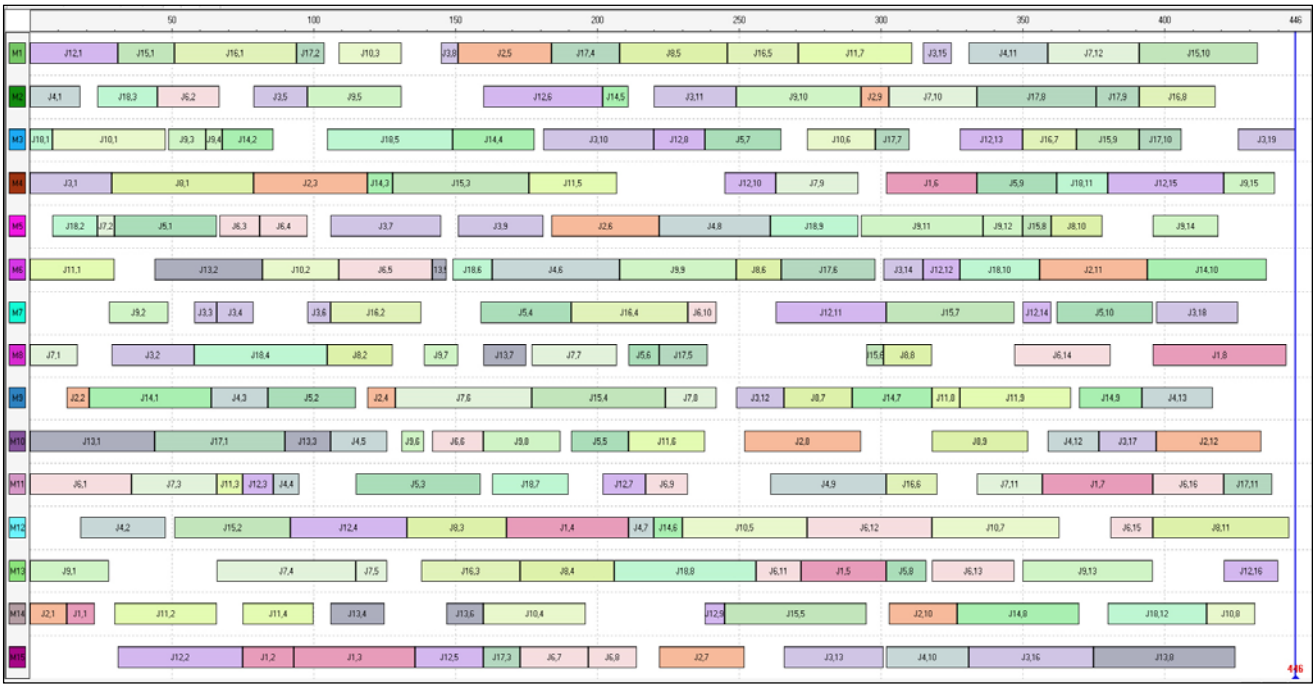

Fig. 13. Gantt chart of Problem 24 in Experiment 3 (Makespan=446)

VI. Real-World Case Study

A. *Description of Case*

The proposed GAVNS algorithm has been applied on a real-world case. To evaluate its performance, two instances are generated according to the practical situations of one machine tool company in China. One main product of this company is the packaging machine (Fig. 14) which is one of the commonly used modules in an automated production line.

In this company, its non-standard equipment production workshop including 13 machines is used to produce 9 kinds of parts in the packaging machine. In the practical production, 9 kinds of parts are processed at the same time. The different parts have different process plans. And for each part, there are a variety of flexible process plans, including machine flexibility, sequencing flexibility and processing flexibility. One example (Part 3) with its flexible process planning information are shown in Fig. 15 and Table VI. Only scheduling cannot improve its productivity greatly. Therefore, this workshop should consider the process planning and scheduling simultaneously. It is a typical IPPS problem.

According to the real-world production situation, two instances containing 9 jobs (contain different parts) and 17 jobs are generated respectively. The details are given in Table VII.

TABLE VI
FLEXIBLE PROCESS PLANNING INFORMATION OF PART 3

Features	Features Description	Alternative Operations	Operation Description	Alternative Machines	Processing time	Precedence constraints
F1		O1-O2	Milling	M1, M2, M3	60, 60, 60	Before all features
			Grinding	M10, M11	60, 60	
F2	Φ80 Hole	O3-O4	Lathing	M5, M6	60, 60	Before F3
			Grinding	M12	40	
F3	Φ33 Hole	O5-O6	Boring	M7	40	Before F4, F5, F6
			Grinding	M12	40	
		O7-O8	Drilling	M4	30	
			Grinding	M12	40	
F4	M6 Threaded holes	O9	Drilling	M1, M2, M3	40, 40, 40	Before F7
		O10-O11	Drilling	M8, M9	60, 60, 60	
			Tapping	M13	20	
F5	M6 Threaded holes	O12	Drilling	M1, M2, M3	20, 20, 20	Before F8
		O13-O14	Drilling	M8, M9	20, 20	
			Tapping	M13	10	
F6	Hole	O15	Milling	M1, M2, M3	50, 50, 50	
F7	M8 Threaded holes	O16	Drilling	M1, M2, M3	30, 30, 30	
		O17-O18	Drilling	M8, M9	30, 30	
			Tapping	M13	10	
F8	M6 Hole	O19	Drilling	M1, M2, M3	10, 10, 10	
		O20-O21	Drilling	M8, M9	10, 10	
			Tapping	M13	5	

B. *Case Results*

Table VIII shows the case results and the comparisons with the simple GA algorithm. It is clear from Table VIII that the results of GAVNS algorithm better than those of the simple GA. The computational results show that GAVNS can obtain the optimal or near-optimal solutions for the real-world IPPS problems effectively.

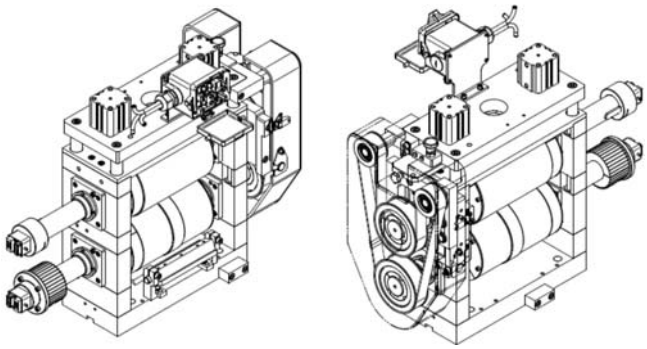

Fig. 14. Packaging Machine

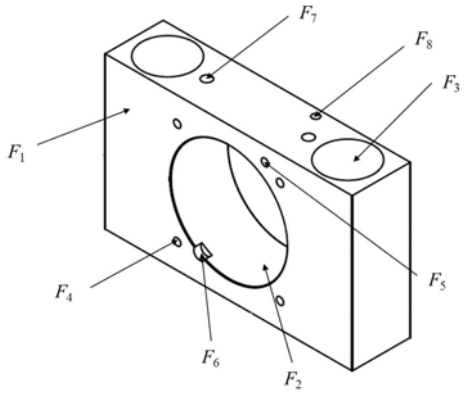

Fig. 15. Part 3 with 8 features

TABLE VII
THE INFORMATION OF 2 REAL-WORLD PROBLEM INSTANCES

Instance	Job No.	Part No.	Job No.	Part No.	Job No.	Part No.
1	Job 1	Part 1	Job 4	Part 4	Job 7	Part 7
(9 jobs)	Job 2	Part 2	Job 5	Part 5	Job 8	Part 8
	Job 3	Part 3	Job 6	Part 6	Job 9	Part 9
2	Job 1	Part 1	Job 7	Part 3	Job 13	Part 6
(17 jobs)	Job 2	Part 2	Job 8	Part 4	Job 14	Part 7
	Job 3	Part 2	Job 9	Part 4	Job 15	Part 7
	Job 4	Part 2	Job 10	Part 5	Job 16	Part 8
	Job 5	Part 2	Job 11	Part 5	Job 17	Part 9
	Job 6	Part 3	Job 12	Part 6		

TABLE VIII
THE CASE RESULTS

Solution methods	Simple GA	GAVNS
Instance 1	465	**395**
Instance 2	512	**455**

VII. CONCLUSIONS AND FUTURE RESEARCHES

This research considers the machine flexibility, sequencing flexibility and processing flexibility of process plans simultaneously. A hybrid GAVNS algorithm which is mixed by the GA and VNS is proposed to optimize the IPPS problem. Three experimental studies including 37 well-known benchmark problems have been employed to test the approach, and the comparisons with the state-of-the-art methods have been given. The results show that the proposed method is very effective for the IPPS problem. The proposed method finds the new solutions for some well-known benchmark problems. The proposed algorithm has also been applied on a real-world manufacturing case coming from a non-standard equipment production workshop of a machine tool company in China to demonstrate its applicability, and the result shows that it also can solve this real-world case very well.

As future work, IPPS with multi-objective can be a significant research topic for the extension of this work [49]. Considering other objectives, such as the energy consumption in scheduling [50-51] to solve other large scale real-world problems can be another future research direction.

REFERENCES

[1] X. Li, L. Gao, and X. Shao, "An active learning genetic algorithm for integrated process planning and scheduling," *Expert Syst. Appl.*, vol. 39, no. 8, pp. 6683-6691, 2012.

[2] X. Li, L. Gao, and X. Wen, "Application of an efficient modified particle swarm optimization algorithm for process planning," *Int. J. Adv. Manuf. Tech.*, vol. 67, no. 5-8, pp. 1355-1369, 2013.

[3] S. Zhang, and T. Wong, "Integrated process planning and scheduling: an enhanced ant colony optimization heuristic with parameter tuning," *J. Intell. Manuf.*, vol. 29, pp. 585-601, 2018.

[4] S. Wang, and L. Wang, "An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem," *IEEE Trans. Syst. Man Cybern. -Syst.*, vol. 46, no. 1, pp.139-149, 2016.

[5] Q. Pan, L. Wang, H. Sang, J. Li, and M. Liu, "A high performing memetic algorithm for the flowshop scheduling problem with blocking," *IEEE T. Autom. Sci. Eng.*, vol. 10, no. 3, pp. 741-756, 2013.

[6] X. Zheng, and L. Wang, "A collaborative multiobjective fruit fly optimization algorithm for the resource constrained unrelated parallel machine green scheduling problem," *IEEE Trans. Syst. Man Cybern. -Syst.*, vol. 48, no. 5, pp. 790-800, 2018.

[7] J. Wang, and L. Wang, "A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop," *IEEE Trans. Syst. Man Cybern. -Syst.*, DOI: 10.1109/TSMC.2017.2788879, 2018.

[8] H. Ge, L. Sun, Y. Liang, and F. Qian, "An effective PSO and AIS-based hybrid intelligent algorithm for job-shop scheduling," *IEEE Trans. Syst. Man Cybern. Part A-Syst. Hum.*, vol. 38, no. 2, pp. 358-368, 2008.

[9] G. Mejia, J. Villalobos, and C. Montoya, "Petri nets and deadlock-free scheduling of open shop manufacturing systems," *IEEE Trans. Syst. Man Cybern. -Syst.*, vol. 48, no. 6, pp. 1017-1028, 2018.

[10] Y. Han, D. Gong, Y. Jin, and Q. Pan, "Evolutionary multiobjective blocking lot-streaming flow shop scheduling with machine breakdowns," *IEEE T. Cybern.*, DOI: 10.1109/TCYB.2017.2771213, 2018.

[11] L. Shi, and Y. Pan. "An efficient search method for job-shop scheduling problems," *IEEE T. Autom. Sci. Eng.*, vol. 2, no. 1, pp. 73-77, 2005.

[12] K. Gao, F. Yang, M. Zhou, Q. Pan, and P. Suganthan, "Flexible job-shop rescheduling for new job insertion by using discrete jaya algorithm," *IEEE T. Cybern.*, DOI: 10.1109/TCYB.2018.2817240, 2018.

[13] X. Shao, X. Li, L. Gao, and C. Zhang, "Integration of process planning and scheduling -- A modified genetic algorithm-based approach," *Comput. Oper. Res.*, vol. 36, no. 6, pp. 2082-2096, 2009.

[14] M. Gen, W. Zhang, L. Lin, and Y. Yun, "Recent advances in hybrid evolutionary algorithms for multiobjective manufacturing scheduling," *Comput. Ind. Eng.*, vol. 112, pp. 616-633, 2017.

[15] L. Qiao, and S. Lv, "An improved genetic algorithm for integrated process planning and scheduling," *Int. J. Adv. Manuf. Tech.*, vol. 58, no. 5-8, pp. 727-740, 2012.

[16] O. Sobeyko, and L. Monch, "Integrated process planning and scheduling for large-scale flexible job shops using metaheuristics," *Int. J. Prod. Res.*, vol. 55, no. 2, pp. 392-409, 2017.

[17] M. Yu, Y. Zhang, K. Chen, and D. Zhang, "Integration of process planning and scheduling using hybrid GA/PSO algorithm". *Int. J. Adv. Manuf. Tech.*, vol. 78, pp. 583-592, 2015.

[18] H. Xia, X. Li, and L. Gao, "A hybrid genetic algorithm with variable neighborhood search for dynamic integrated process planning and scheduling," *Comput. Ind. Eng.*, vol. 102, pp. 99-112, 2016.

[19] G. Chryssolouris, S. Chan, and W. Cobb, "Decision making on the factory floor: an Integrated approach to process planning and scheduling," *Robot. Comput. Integr. Manuf.*, vol. 1, no. 3-4, pp. 315-319, 1984.

[20] W. Tan and B. Khoshnevis, "Integration of process planning and scheduling – a review," *J. Intell. Manuf.*, vol. 11, no. 1, pp. 51-63, 2000.

[21] Y. K. Kim, K. Park, and J. Ko, "A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling," *Comput. Oper. Res.*, vol. 30, no. 8, pp. 1151-1171, 2003.

[22] X. Li, C. Zhang, L. Gao, W. Li, and X. Shao, "An agent-based approach for integrated process planning and scheduling," *Expert Syst. Appl.*, vol. 37, no. 2, pp. 1256-1264, 2010.

[23] X. Li, L. Gao, C. Zhang, and X. Shao, "A review on integrated process planning and scheduling," *Int. J. Manuf. Res.*, vol. 5, no. 2, pp. 161-180, 2010.

[24] X. Li, L. Gao, X. Shao, C. Zhang, and C. Wang, "Mathematical modeling and evolutionary algorithm-based approach for integrated process planning and scheduling," *Comput. Oper. Res.*, vol. 37, no. 4, pp. 656-667, 2010.

[25] L. Zhang, and T. Wong, "An object-coding genetic algorithm for integrated process planning and scheduling," *Eur. J. Oper. Res.*, vol. 244, pp. 434-444, 2015.

[26] G. Luo, X. Wen, H. Li, W. Ming, and G. Xie, "An effective multi-objective genetic algorithm based on immune principle and external archive for multi-objective integrated process planning and scheduling," *Int. J. Adv. Manuf. Tech.*, vol. 91, pp. 3145-3158, 2017.

[27] E. Shokouhi, "Integrated multi-objective process planning and flexible job shop scheduling considering precedence constraints," *Prod. Manuf. Res.*, vol. 6, no. 1, pp. 61-89, 2018.

[28] W. Shen, L. Wang, and Q. Hao, "Agent-Based Distributed Manufacturing Process Planning and Scheduling: A State-of-the-Art Survey," *IEEE Trans. Syst. Man Cybern. Part C-Appl. Rev.*, vol. 36, no. 4, pp. 563-577, 2006.

[29] V.Manupati, G. Putnik, M. Tiwari, P. Avila, and M. Cunha, "Integration of process planning and scheduling using mobile-agent based approach in a networked manufacturing environment," *Comput. Ind. Eng.*, vol. 94, pp. 63-73, 2016.

[30] T. N. Wong, S. Zhang, G. Wang, and L. Zhang, "Integrated process planning and scheduling–multi-agent system with two-stage ant colony optimisation algorithm," *Int. J. Prod. Res.*, vol. 50, no. 21, pp. 6188-6201, 2012.

[31] N. Mishra, A. Singh, S. Kumari, and K. Govindan, "Cloud-based multi-agent architecture for effective planning and scheduling of distributed manufacturing," *Int. J. Prod. Res.*, vol. 54, no. 23, pp. 7115-7128, 2016.

[32] M. Petrovic, N. Vukovic, M. Mitic, and Z. Miljkovic, "Integration of process planning and scheduling using chaotic particle swarm optimization algorithm," *Expert Syst. Appl.*, vol. 64, pp. 569-588, 2016.

[33] Z. Miljkovic, and M. Petrovic, "Application of modified multi-objective particle swarm optimisation algorithm for flexible process planning problem," *Int. J. Comput. Integr. Manuf.*, vol. 30, No. 2-3, pp. 271-291, 2017.

[34] L. Zhang, and T. Wong, "Solving integrated process planning and scheduling problem with constructive meta-heuristics," *Inf. Sci.*, vol. 340-341, pp. 1-16, 2016.

[35] B. Zhao, J. Gao, K. Chen, and K. Guo, "Two-generation Pareto ant colony algorithm for multi-objective job shop scheduling problem with alternative process plans and unrelated parallel machines," *J. Intell. Manuf.*, vol. 29, pp. 93-108, 2018.

[36] W. Li and C. McMahon, "A simulated annealing – based optimization approach for integrated process planning and scheduling," *Int. J. Comput. Integr. Manuf.*, vol. 20, no. 1, pp. 80-95, 2007.

[37] F. Chan, V. Kumar, and M. Tiwari, "The relevance of outsourcing and leagile strategies in performance optimization of an integrated process planning and scheduling model," *Int. J. Prod. Res.*, vol. 47, no. 1, pp. 119-142, 2009.

[38] X. Li, X. Shao, L. Gao, and W. Qian, "An effective hybrid algorithm for integrated process planning and scheduling," *Int. J. Prod. Econ.*, vol. 126, no. 2, pp. 289-298, 2010.

[39] M. Naseri, and A. Afshari, "A hybrid genetic algorithm for integrated process planning and scheduling problem with precedence constraints," *Int. J. Adv. Manuf. Tech.*, vol. 59, no. 1-4, pp. 273-287, 2012.

[40] N. Mladenovic, and P. Hansen, "Variable neighborhood search," *Comput. Oper. Res.*, vol. 24, no. 11, pp. 1097-1100, 1997.

[41] E. Nowicki, and C. Smutnicki, "A fast taboo search algorithm for the job shop scheduling problem," *Manage. Sci.*, vol. 42, no. 6, pp. 797-813, 1996.

[42] C. Moon, Y. Lee, C. Jeong, and Y. Yun, "Integrated process planning and scheduling in a supply chain," *Comput. Ind. Eng.*, vol. 54, no. 4, pp. 1048-1061, 2008

[43] F. Chan, V. Kumar, and M. Tiwari, "Optimizing the performance of an integrated process planning and scheduling problem: an AIS-FLC based approach," in *Proc. CIS*, pp. 1-8, 2006.

[44] C. Leung, T. Wong, K. Maka, and R. Fung, "Integrated Process planning and scheduling by an agent-based ant colony optimization," *Comput. Ind. Eng.*, vol. 59, no. 1, pp. 166-180, 2010.
[45] S. Lv, W. Liu, "A cross-entropy-based approach for joint process plan selection and scheduling optimization," *Proc. Inst. Mech. Eng. Part B-J. Eng. Manuf.*, vol. 230, no. 8, pp. 1525-1536, 2016.
[46] I. Chaudhry, and M. Usman, "Integrated process planning and scheduling using genetic algorithms," *Teh. Vjesn.*, vol. 24, no. 5, pp. 1401-1409, 2017.
[47] Y. Kim, "A set of data for the integration of process planning and job shop scheduling," Available at: http://syslab.chonnam.ac.kr/links/data-pp&s.doc.
[48] K. Lian, C. Zhang, L. Gao, and X. Li, "Integrated process planning and scheduling using an imperialist competitive algorithm," *Int. J. Prod. Res.*, vol. 50, no. 15, pp. 4326-4343, 2012.
[49] Y. Yuan, H. Xu, "Multiobjective flexible job shop scheduling using memetic algorithms," *IEEE T. Autom. Sci. Eng.*, vol. 12, no. 1, pp. 336-353, 2015.
[50] G. Chen, L. Zhang, J. Arinez, and S. Biller, "Energy-efficient production systems through schedule-based operations," *IEEE T. Autom. Sci. Eng.*, vol. 10, no. 1, pp. 27-37, 2013.
[51] X. Li, C. Lu, L. Gao, S. Xiao, and L. Wen, "An effective multiobjective algorithm for energy-efficient scheduling in a real-life welding shop," *IEEE Trans. Ind. Inform.*, DOI: 10.1109/TII.2018.2843441, 2018.

Xinyu Li received the Ph.D. degree in industrial engineering from Huazhong University of Science and Technology (HUST), China, 2009.

He is an Associate Professor of the Department of Industrial & Manufacturing Systems Engineering, State Key Laboratory of Digital Manufacturing Equipment & Technology, School of Mechanical Science & Engineering, HUST. He had published more than 70 refereed papers. His research interests include intelligent scheduling, machine learning etc.

Liang Gao (M'08) received the Ph.D. degree in mechatronic engineering from Huazhong University of Science and Technology (HUST), China, 2002.

He is a Professor of the Department of Industrial & Manufacturing System Engineering, State Key Laboratory of Digital Manufacturing Equipment & Technology, School of Mechanical Science & Engineering, HUST. He had published more than 170 refereed papers. His research interests include operations research and optimization, scheduling, big data and machine learning etc. Prof. GAO is an Associate Editor of Swarm and Evolutionary Computation, Journal of Industrial and Production Engineering and Swarm Intelligence and Numerical Method. He is an editor board member of European Journal of Industrial Engineering.

Quan-ke Pan received the B.Sc. degree and the Ph.D. degree from Nanjing university of Aeronautics and Astronautics, Nanjing, China, in 1993 and 2003, respectively.

From 2003 to 2011, he was with School of Computer Science Department, Liaocheng University, where he became a Full Professor in 2006. From 2011 to 2014, he was with State Key Laboratory of Synthetical Automation for Process Industries (Northeastern University), Shenyang, China. He has been with State Key Laboratory of Digital Manufacturing and Equipment Technology (Huazhong University of Science & Technology) since 2014. His current research interests include intelligent optimization and scheduling. He has authored one academic book and more than 150 refereed papers.

Liang Wan received the Master degree in Industrial Engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, 2014.

His research interests include scheduling, etc.

Kuo-Ming Chao's research interests include the areas of intelligent agents, service-oriented computing, cloud computing and big data, as well as their applications such as energy efficiency management and green manufacturing etc.

He has over 200 refereed publications in books, journals, and conference proceedings. Prof. Chao is a co-founder and Editors-in-Chief of Service-Oriented Computing and Applications: A Springer Journal to promote Service-Oriented Computing. He is member of editorial broads for various international journals. In addition, he is involved a number of EU-funded projects as coordinator or work pack leader. Prof. Chao has served several international conferences by taking different responsibilities such as general, programme and track chairs.