A priority-based heuristic algorithm (PBHA) for optimizing integrated process planning and scheduling problem

Muhammad Farhan Ausaf¹, Liang Gao¹*, Xinyu Li¹ and Ghiath Al Aqel¹

Abstract: Process planning and scheduling are two important components of a manufacturing setup. It is important to integrate them to achieve better global optimality and improved system performance. To find optimal solutions for integrated process planning and scheduling (IPPS) problem, numerous algorithm-based approaches exist. Most of these approaches try to use existing meta-heuristic algorithms for solving the IPPS problem. Although these approaches have been shown to be effective in optimizing the IPPS problem, there is still room for improvement in terms of quality of solution and algorithm efficiency, especially for more complicated problems. Dispatching rules have been successfully utilized for solving complicated scheduling problems, but haven't been considered extensively for the IPPS problem. This approach incorporates dispatching rules with the concept of prioritizing jobs, in an algorithm called priority-based heuristic algorithm (PBHA). PBHA tries to establish job and machine priority for selecting operations. Priority assignment and a set of dispatching rules are simultaneously used to generate both the process plans and schedules for all jobs and machines. The algorithm was tested for a series of benchmark problems. The proposed algorithm was able to achieve superior results for most complex problems presented in recent literature while utilizing lesser computational resources.

ABOUT THE AUTHORS
Muhammad Farhan Ausaf is a PhD student at Huazhong University of Science and Technology (HUST). His research interests include optimization, intelligent algorithms and scheduling.

Liang Gao is a professor and head of the Department of Industrial & Manufacturing System Engineering, Huazhong University of Science and Technology. His research interests include operations research and optimization, scheduling, etc. He has published more than 40 journal papers.

Xinyu Li is a lecturer in the Department of Industrial & Manufacturing Systems Engineering, School of Mechanical Science and Engineering, HUST. His research interests include integrated process planning and scheduling, flexible job-shop scheduling and intelligent algorithm.

Ghiath Al Aqel is a PhD student at Huazhong University of Science and Technology (HUST). His research interests include optimization, intelligent algorithms and scheduling.

PUBLIC INTEREST STATEMENT
In a manufacturing setup, the key objective is to get maximum productivity from available resources. In a job-shop environment, multiple products are being processed simultaneously. Selecting the right processes (process planning) and the right order to process those (scheduling) are vital for improving productivity of a manufacturing system. Since the ultimate goal is to improve the overall productivity of a manufacturing system, it makes sense to treat process planning and scheduling simultaneously. Since both process planning and scheduling problems are very complex problems, even for moderate integrated process planning and scheduling (IPPS) problems the number of possible solutions can be very high. It is impossible to manually explore these solutions and pick the best one. In this regard, algorithms are used to find the optimal solution to the IPPS problem. In this research, a new algorithm called a priority-based heuristic algorithm (PBHA) is presented for finding the best (optimal) solutions.
1. Introduction

Process planning can be defined as the act of selection and sequencing of different processes and operations to transform a set of given raw materials into a finished component (Scallan, 2003). Scheduling, on the other hand, is related to the assignment of all the operations in every job to the available machines, ensuring that all the precedence constraints defined in the process plan are satisfied (Sugimura, Hino, & Mriwaki, 2001). Scheduling can be classified based on the job arrival patterns, machines in a shop and the flow patterns in the shop (Conway, Maxwell, & Miller 1967). The scheduling problem considered in this work is the flexible job-shop scheduling Problem (FJSP). FJSP is the extension of the classic job-shop scheduling problem (JSP). The main variation in FJSP is that unlike JSP an operation can have more than one feasible machine, thus making the problem more complicated.

Traditionally, process planning and scheduling were treated as separate entities. Over the past few decades, efforts have been directed towards having manufacturing systems, where all components can be combined to work as a single cohesive unit. The integration of process planning and scheduling is a major task in achieving this goal. The traditional approach, in which process planning was performed first and using those fixed process plans, scheduling followed, has numerous drawbacks as pointed out in the literature. These drawbacks include unbalanced resource utilization, infeasible and unrealistic process plans, uncoordinated and isolated optimization.

To rectify these drawbacks, the concept of integration of process planning and scheduling was first proposed by Chryssolouris and Chan (1985) and Chryssolouris, Chan, and Cobb (1984). Since then, integrated process planning and scheduling (IPPS) has been an active topic among researchers. During this time, extensive research has been done on its various aspects including the model and framework (Kumar & Rajotia, 2006; Liu, Bai, & Zhang, 2004; Mamalis, Malagardis, & Kambouris, 1996; Wang, Song, & Shen, 2007; Zhang, Gao, & Chan, 2003), system building (Grabowik, Kalinowski, & Monica, 2005; Yang, Parsaei, & Leep, 2001) and optimization of the IPPS problem using algorithms (Cai, Wag, & Feng, 2009; Kim, Park, & Ko, 2003; Lee & Kim, 2001; Li & McMahon, 2007; Lim & Zhang, 2004; Shao, Li, Gao, & Zhang, 2009; Zhao, 2006). Comprehensive details of all the recent work can be found in Li, Gao, Zhang, and Shao (2010) and Phanden, Jain, and Verma (2011) and more details on earlier work can be found in Tan and Khoshnevis (2000).

IPPS is a non-polynomial hard problem and the use of approximation algorithm, especially heuristic algorithms like Genetic Algorithm (GA) (Choi & Park, 2006), and Simulated Annealing (SA) (Palmer, 1996) provides the best way to find an optimal solution. During the past decade, most of the research in optimization of IPPS has been focused on finding more effective algorithms (Guo, Li, & Mileham, 2009; Kim et al., 2003; Li & McMahon, 2008), modifying existing algorithms (Cai et al., 2009; Lee & Kim, 2001; Shao et al., 2009) or producing hybrid algorithms (Amin-Naseri & Afshari, 2012; Li, Shao, Gao, & Qian, 2010; Zhao, 2010). Because of their agility, flexibility and ease of use dispatching rules are widely used for solving scheduling problems (Chiang & Fu, 2007), but not much effort has been dedicated to the use of dispatching rules in the optimization of the IPPS problem. This novel approach tries to incorporate the use of dispatching rules with the concept of prioritization to solve the IPPS problem. The result is a simple, fast and efficient algorithm which can effectively solve the IPPS problem.

The remainder of this paper is organized as follows. Section 2 gives a brief account of the related research work. Different aspects of IPPS considered in this research are discussed in Section 3. The detailed working of the algorithm is explained in Section 4. The experiments and their results are discussed in Section 5 and the paper is concluded in Section 6.
2. Related work
A lot of work has been done on IPPS. Since this research is related to finding an efficient algorithm for the optimization of IPPS problem, only some related previous work is presented.

With the improvement in the computation capabilities of modern computers, algorithm-based approach for IPPS has gained increasing interest. Initially, simple algorithms, mostly based on the branch and bound scheme (Brucker, Jurisch, & Sievers, 1994) had reasonable success for smaller problems. The first instance of the use of intelligent algorithms for optimization of IPPS problem can be traced back to Palmer (1996) who used SA in conjunction with three different configurations, i.e. sequence change on a machine, sequence change within a job and alternate methods of operations. Functions like tardiness, mean flow time, makespan and a combined function of mean flow time and tardiness were considered. A comparison was made with dispatching rules and the results suggested that the solution quality for SA was much better than dispatching rules.

Other researchers have also tried to incorporate different variations of SA for the IPPS problem. Li and McMahon (2007) used SA to solve for the IPPS in a job-shop environment. Processing, operation sequence and scheduling flexibilities were considered. The algorithm was evaluated for optimization of makespan, balanced machine utilization, job tardiness and manufacturing cost. The results were compared with GA, particle swarm optimization (PSO) and tabu search (TS) algorithms and the authors concluded that this algorithm yielded satisfactory results. Shukla, Tiwari, and Son (2008) used a hybrid TS-SA algorithm with a bidding based multi-agent system (MAS) to find an optimal process plan and schedule. Chan, Kumar, and Tiwari (2009) proposed an enhanced swift converging simulated annealing (ESCSA) algorithm. The proposed algorithm was compared with other optimization algorithms like GA, TS, SA and TS-SA and it was found that this algorithm outperforms all.

A lot of work has been done on the application of GA for optimizing IPPS. The first contributors in the regard were Morad and Zalzala (1999), who considered different aspects of the problem like processing time, alternate machine, machine tolerances and processing cost. Lee and Kim (2001) used a GA based simulation method for IPPS. GA was used to generate combination of process plans and the near-optimal process plan combination is outputted prior to execution on shop floor. The performance measures were makespan and lateness based on shortest processing time (SPT) and earliest due date (EDD) dispatching rules. Compared to the random selection of the process plan, they observed 20% reduction in makespan.

Moon, Kim, and Hur (2002) explored IPPS for multi-plant supply chain using GA. A mathematical model was formulated to minimize tardiness. A topological short technique (TST) was used to obtain all flexible sequences. The authors concluded that their GA approach was more efficient both in terms of computational time and problem size compared to TS. Zhao (2004) used fuzzy logic in conjunction with a GA-based approach for IPPS in a job-shop environment. The fuzzy logic toolbox of MATLAB was the basis for this fuzzy inference used to select alternative machines. GA was used to balance the load for all machines. The objectives considered were to minimize makespan, number of rejects and processing costs.

Choi and Park (2006) also used a GA based method for IPPS. For an integrated manufacturing environment, they considered alternative machines and alternative operation sequences and considered minimization of makespan as the objective function. The authors concluded that the proposed approach shows the possibility of improving makespan. Li, Gao, Zhang, Zhang, and Shao (2008) proposed an approach based on GA for IPPS. Their genetic representation consisted of two-part chromosomes in which the first was used to store the alternate process plan, while the second to store the scheduling plan. The objective was to minimize makespan and they concluded that the value of makespan was improved by considering this integration model as compared to the traditional without integration approach.
Shao et al. (2009) used modified GA in a simulation approach which was aimed at synthesizing integration methodology of nonlinear approach (NLA) and distributed approach (DA). The objective was to minimize makespan and a combined objective of makespan and balanced level of machine utilization. They concluded that their approach was better than hierarchal approach. Li, Shao, et al. (2010) proposed a hybrid algorithm which combined the advantages of GA and TS, to solve for the IPPS problem. The three-part chromosomes had information for alternate process plans, scheduling plan as well have available machines. The authors concluded that this algorithm was capable of effectively solving the aforementioned problem.

Chaudhry (2012) presented a spreadsheet based GA to solve IPPS problem. The shop model was built in Microsoft Excel spreadsheet to find the optimized process plan and corresponding schedule. The author suggested the proposed approach to be general purpose and capable of optimizing any objective function without changing the model or the GA routine. Phanden, Jain, and Verma, (2013) proposed an approach to quickly integrate process planning and scheduling in companies with existing process planning and scheduling departments. They used a GA based simulation approach and the integration methodology is accessed on mean tardiness and makespan and compared with the hierarchal approach concluding that the proposed integration approach performs better than hierarchal approach.

Naseri and Ahmed (2012) proposed a hybrid GA. They employed a problem-specific genetic operator to enhance the global search power of GA. A local search procedure was also incorporated into the GA to improve its performance. They considered precedence relationship among job operations. They concluded the proposed algorithm was efficient in finding optimal or near-optimal solutions. Lihong and Shengping (2012) proposed an improved GA (IGA) for the problem. Their algorithm applied new initial selection method for process plans, new genetic representations for combining scheduling and process plans and genetic operator method. Using makespan and mean flow time, a comparison was made with other existing algorithms for benchmark problems. The authors concluded that their proposed approach has achieved significant improvement in minimizing makespan and obtained good results for the improvement of mean flow time.

Weintraub (1999) utilized TS for IPPS. They proposed a procedure for scheduling jobs while considering alternative process plans in a large-scale manufacturing job shop. The objective was to minimize manufacturing cost while satisfying due dates. They concluded that this approach can greatly help achieve the goal of satisfying due dates under varying shop conditions. They also concluded that having alternative operations can improve scheduling more as compared to having alternative sequencing.

Chan, Kumar, and Tiwari, (2006) proposed an artificial immune system (AIS)-based algorithm inherited with fuzzy logic controller (FLC). While considering manufacturing system with alternate operation sequencing, alternate machines for operations and precedence relationship among operations their proposed algorithm can handle multiple orders involving outsourcing strategy. The objective function was to minimize makespan while considering customer order due dates. The algorithm was tested for five machines including one outsourcing machine and the authors concluded that outsourcing strategy can be beneficial when total transportation time for outsourcing is less than the waiting time for parts.

Zhao (2006) used a fuzzy inference system in selection of alternate machines for IPPS. They used PSO algorithm to balance the load on each machine. They used objectives like integrating production capability and load balancing to be optimized individually and simultaneously. From the simulation, they concluded that this approach shows promising results. Moreover, Zhao (2010) proposed an IPPS applicable to Holonic Manufacturing System (HMS) using a hybrid PSO and evolution algorithm to balance the load for all machines. Guo et al. (2009) also proposed the utilization of PSO algorithm and replanning method to cater machine breakdowns and new order arrival. Their work showed that PSO algorithm was computationally more efficient than both GA and SA.
Kim et al. (2003) proposed a symbiotic evolutionary algorithm (SEA) to simultaneously deal with process planning and job-shop scheduling in flexible manufacturing system (FMS). The basis of their algorithms was on the fact that compared to single search for the entire solution, a parallel search for different pieces of solution was more efficient. In a comprehensive study, they considered operation flexibility, sequencing flexibility and processing flexibility during process planning. Two types of optimization criteria minimizing makespan and minimizing mean flow time were considered. SEA was tested on a 24 test-bed problem set and solutions better than those of existing corporate co-evolutionary genetic algorithm (CCGA) as well as hierarchal approach (HA) were found.

Lian, Zhang, Gao, and Li (2011) proposed an imperialist competitive algorithm (ICA) to address the IPPS problem. They considered extended operation based representation scheme to include information related to various types of flexibilities related to process planning and scheduling for a job-shop environment. Performance of the proposed ICA was compared with other existing algorithms like HA, evolutionary algorithm (EA) and CCGA and the authors concluded that ICA can effectively solve the IPPS problem.

Wong, Zhang, Wang, and Zhang (2012) presented a two-stage ant colony optimization (ACO) algorithm implemented in a MAS to accomplish IPPS in job shop-type flexible manufacturing environments. They concluded that this algorithm is effective and efficient in generating feasible solutions for IPPS problems with moderate complexity.

For the IPPS optimization problem, the work done by researchers has been summarized above. This research suggests that heuristic algorithms have shown promising results in optimizing the IPPS problem. The main idea of an algorithm based approach is to develop algorithms which can effectively and efficiently explore the search space. Researchers in this regard have not focused on using specialized algorithm for IPPS problem. Dispatching rules, which have been successfully employed for solving complicated scheduling problems, have not been sufficiently explored for optimization of IPPS problem. In this work a heuristic algorithm is proposed, which is based on how things are prioritized in typical job shops. The proposed algorithm uses a set of dispatching rules in conjunction with a priority assignment mechanism to optimize the IPPS problem.

3. IPPS problem description
A lot of optimization criteria have been considered for the IPPS problem, but the most common is makespan. The IPPS problem for optimization of makespan can be defined as in Tan and Khoshnevis (2000): Given “n” jobs are to be processed on “M” machines, with jobs having possible alternative operation sequences and their operations requiring alternative machining resources, the objective is to optimize the criterion of minimizing makespan by selecting a suitable machine for each operation, an operation sequence for each job along with the complete schedule while satisfying all precedence constraints.

The objective of minimizing makespan can be mathematically expressed as:

\[
\text{Minimize } MS = \max(c_i) \quad \forall i \in [1, n]
\]  

(1)

where MS is the makespan, \( c_i \) is the total time to complete job “i” and “n” is the total number of jobs.

The total time to complete a job is the sum of individual times for each operation. Mathematically,

\[
c_i = \sum_{j=1}^{o_i} t_{jk} \forall k \in [1, M]
\]  

(2)

where \( t_{jk} \) is the time required to complete the jth operation on kth machine, and “o_i” is the total number of operations for job “i”.
Although makespan is the optimization criterion for only scheduling, it is widely used as an optimization criterion for optimization of the IPPS problem. The main difference here is that scheduling does not follow a predefined fixed process plan, rather the process plan is generated or selected from a pool, in conjunction with the scheduling, keeping in view a broader prospective of the optimization of the entire manufacturing system. The process plan finally generated or selected may not be the best one when considering process planning alone, but will be optimized for the desired final objective, which is makespan in this case.

3.1. Assumptions

The IPPS problem defined above will be subjected to the following constraints:

1. All jobs are independent of each other.
2. All machines are available at the beginning and processing of all jobs can be started immediately.
3. Job preemption is not allowed.
4. Each machine can only handle one operation at a time.
5. Multiple operations of the same job cannot be processed simultaneously.
6. Setup time is not dependent on the sequence of operations and is assumed to be included in the processing time.
7. The transportation time for jobs between machines is negligible compared to the processing time and can be ignored.
8. Machine breakdown and other interruptions on the shop floor are ignored.

3.2. Problem flexibility and its representation

For the IPPS problem, three kinds of flexibilities (Benjaafar & Ramakrishnan, 1996; Kim et al., 2003) are taken into consideration. Operation flexibility (OF), states that an operation can be performed on more than one machine, the processing and setup time as well as the cost of machining can vary for different machines. Sequencing Flexibility (SF) provides different combination of sequences, in which operations in a job can be performed. Processing flexibility (PF) is related to having alternative manufacturing options for jobs, i.e. same features in a job can be generated using alternate operations.

For representing these flexibilities, popular techniques used are the Petri networks (Lee & Dicesare, 1994), operation relationship graphs (Yang, Qiao, & Jiang, 1998) and ANDOR networks (Ho & Moodie, 1996). In this paper, a new set based representation, which is derived from the ANDOR networks, is...
used. This representation, although not as visual as its predecessor, has the advantage of being very compact and easy to code.

An ANDOR graph for a job taken from the data used in Kim et al. (2003) and its corresponding set based representation are shown in Figure 1.

The dummy start and finish nodes are eliminated. The independent groups of operations called chains (detailed description in the next section) are represented by curly brackets “{ }”. Within the chains, square brackets “[ ]” with “&” and “^” notations are used to represent the operations connected by AND and OR junctions, respectively.

4. Priority-based heuristic algorithm
To better understand the working of PBHA, a few important terms and some associated concepts are discussed below

4.1. Chains
Because of the flexibilities considered in this study, it is possible to perform some operations of a job independent of other operations. Consider Figure 1 which shows a job with 20 operations. Here, the sequence of operations 1–4 can be performed independent of the other operation in this job. Thus, we can consider such sequences as sub-jobs, which are referred to as chains in this paper. So, in this example, the job has three chains. Chains can be regarded as a smaller job with one important difference; although operations of different jobs can be processed simultaneously, operations of different chains of the same job cannot be processed simultaneously.

4.2. The concept of priority
All jobs are available for assignment from the beginning and there is no prejudice among jobs. However, to guide the search mechanism in achieving a better and faster solution, certain jobs and their operations will be given more chance of selection as compared to other jobs and operations. A probability based mechanism is used to assign different priorities to different jobs and operations. If the selection of “n” jobs was done at random, each job would have an equal “1/n” chance of being selected, but based on priority assignment now each job may have a greater or lesser chance of selection. Priorities will be assigned to jobs and also to chains in every job.

4.3. Priority assignment for jobs

4.3.1. Number of following operations
Consider the two jobs shown in Figure 2. Job2 has only one operation, while job1 has four operations. There are a total of five possible sequences to assign these jobs to a machine as shown in Figure 2. If the operation selection is done at random, both operation1 and operation2 will have 50% chance of being selected as the first operation, but selection of operation2 as the first operation only accounts for one of the five possible plans. Thus, in order to even out the selection chances of all five plans, operation1 should be given priority over operation2 when selecting the first operation.

Figure 2. Priority assignment.
The priority-based on the number of following operations for each job is calculated as a probability and is based on the relative number of operations in each job. The total number of operations in each job is used to calculate the probability for each using the following relationship:

\[ P_{o_i} = \frac{O_i}{\sum_{i=1}^{n} O_i} \]  

(3)

where \( n \) is the total number of jobs, \( O_i \) is the total number of operations in the \( i \)th job and \( P_{o_i} \) is the corresponding selection probability for the \( i \)th job.

4.3.2. Critical jobs

In planning, there are always some jobs which will influence the objective function more than others. Since makespan has been chosen as the objective function for this study, jobs which have longer machining time are more critical. If the completion time of these jobs can be kept to a minimum, there are more chances of finding a better makespan. The minimum machining time \( T_m \) required for each job is calculated and the job with the largest \( T_m \) is the critical job \( J_c \) and this \( T_m \) is the critical time \( T_c \). This can be demonstrated with the help of Table 1; the problem data have been taken from Meenakshi Sundaram and Fu (1988). This is a problem with five jobs with four operations each. It can be seen from Table 1 that \( T_m \) for job2 is the largest, so job2 will be the critical job and \( T_c \) will be 27 for this case. These critical values are then used to calculate the critical probability for each job using Equation 4, where \( C_i \) is the critical value for the \( i \)th job.

To assign priority, based on the criticality of the jobs, the value of \( T_m \) for each job is compared with \( T_c \). Using Table 2, a critical value is then assigned to each job:

\[ P_{c_i} = \frac{C_i}{\sum_{i=1}^{n} C_i} \]  

(4)

Table 2 for assignment of critical values has been based on the results of experiments performed on problems given in Kim et al. (2003). Three different classifications of intervals for critical values were tested: equally spaced intervals, intervals skewed towards the critical value and intervals skewed away from the critical value. It was found that best results were obtained when the intervals for critical values were kept smaller near the critical value and gradually increased while moving away. Since 24 is the maximum number of jobs considered in all experiments, it was also found that keeping the number of intervals to five was a reasonable option. Increasing the number of intervals increases the computation time with no significant improvement in results. For problems with a smaller number of jobs, it is even acceptable to reduce the number of intervals to 3.

<table>
<thead>
<tr>
<th>Job</th>
<th>Opr1</th>
<th>Opr2</th>
<th>Opr3</th>
<th>Opr4</th>
<th>( T_m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job1</td>
<td>5(M1)</td>
<td>7(M2)</td>
<td>6(M3)</td>
<td>3(M4)</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>3(M2)</td>
<td>4(M5)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Job2</td>
<td>7(M1)</td>
<td>4(M2)</td>
<td>7(M3)</td>
<td>10(M5)</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>6(M3)</td>
<td>7(M4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Job3</td>
<td>4(M1)</td>
<td>5(M4)</td>
<td>6(M4)</td>
<td>4(M5)</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>5(M2)</td>
<td>5(M5)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8(M3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Job4</td>
<td>2(M2)</td>
<td>8(M3)</td>
<td>3(M3)</td>
<td>7(M4)</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>6(M3)</td>
<td>8(M4)</td>
<td>4(M5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Job5</td>
<td>3(M1)</td>
<td>7(M3)</td>
<td>9(M4)</td>
<td>3(M5)</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>5(M3)</td>
<td>6(M5)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
To assign the final probability to each job, a weighted average of both $P_c$ and $P_o$ is calculated using the relationship given in Equation 5:

$$P = wP_c + (1 - w)P_o$$

Here, $w$ is the weight of $P_c$, and it can have any value between 0 and 1. For the current study, it was found that in all experiments the best results were obtained for $w = 1$.

### 4.4. Priority assignment for chains

Since the operations in different chains of the same job cannot be performed simultaneously, all chains in a job are equally critical; thus, the only probability considered for chains is $P_o$. $P_o$ for chains can be calculated using Equation 1 by performing the summation of all the operations in the corresponding chain instead of the job.

### 4.5. Dispatching rules based machine selection

Once an operation has been selected to be assigned to a machine, the next step is to choose an appropriate machine for this operation. The selection of the machine to process the selected operation is done using a set of dispatching rules.

A vast number of dispatching rules are given in the literature and the search for newer better ones is an ongoing effort among researchers. It has been argued that no one dispatching rule can satisfy all optimization criteria; so, there is no magic dispatching rule to solve for different optimization criteria [30]. For the proposed algorithm, five basic dispatching rules, keeping in mind the criterion of makespan, were chosen so as to keep the computation simple and effective. This is not a comprehensive list for the said optimization problem, but since the results for the experiments have been more than satisfactory, further effort had not been put in exploring other dispatching rules. The algorithm itself is flexible to the selection of dispatching rules, so changing this set or increasing the number of dispatching rules will have no effect on the working of the algorithm.

The five dispatching rules used are:

#### 4.5.1. Random selection (RS)

The selection of the machine is done at random from the available pool of machines. This is the simplest way of selection and probably the least efficient one. Random reach may be useful as it will search the complete space unlike other rules which tend to logically eliminate some portion of the search space or give preference to other.

#### 4.5.2. Shortest processing time (SPT)

SPT will select the machine which has the shortest processing time for the operations.

#### 4.5.3. Earliest starting time (EST)

The selection of the machine is done which will provide the earliest starting time for the operations.

---

**Table 2. Critical values for a job**

<table>
<thead>
<tr>
<th>Condition</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_m \geq 0.95T_c$</td>
<td>5</td>
</tr>
<tr>
<td>$0.85T_c \leq T_m &lt; 0.95T_c$</td>
<td>4</td>
</tr>
<tr>
<td>$0.70T_c \leq T_m &lt; 0.85T_c$</td>
<td>3</td>
</tr>
<tr>
<td>$0.50T_c \leq T_m &lt; 0.70T_c$</td>
<td>2</td>
</tr>
<tr>
<td>$T_m &lt; 0.5T_c$</td>
<td>1</td>
</tr>
</tbody>
</table>
4.5.4. Earliest finish time (EFT)
The selection in EFT ensures that the selected machine will give the earliest finish time for the operation.

4.5.5. Least utilized machine (LUM)
The machine which currently has the least number of assigned operations is chosen.

If more than one machine fulfils any criteria, then the selection is done at random.

4.6. Priorities for dispatching rules (cooperation among individuals)
As pointed out by researchers that no dispatching rule can be universally successful, a mechanism inspired by the natural selection is used to gradually increase the utilization of rules which are favoring a particular problem. So instead of having a single solution, a group of solutions collectively called as population is considered. A single solution is termed as an individual. The prioritization mechanism is a source of coordination among individuals.

Initially, the population is divided into groups, equal to the total number of dispatching rules. If “p” is the total number of individuals and “q” is the total number of available dispatching rules, then initially the total individuals in each group will be equal to “pq”. Since five dispatching rules are considered in this study, the population is initially divided into five equal groups of individuals. On each group, one dispatching rule is used for the selection of the machines.

At the end of iteration, the best and the worst groups are identified according to their performance, i.e. based on the average makespan for each group. The individuals are now regrouped so that the dispatching rule which is performing best is given a bigger share of the population by taking it from the portion of the population for the dispatching rule which is performing worst. Suppose the total number of individuals is 10, each group will have 2 individuals. If dispatching rule1 gives the best result, while dispatching rule5 gave the worst result, the group for dispatching rule 1 is increased by 1–3, while that of dispatching rule5 is reduced to 1; the other groups remain unchanged. If \( N_{bg} \) is the number of individuals in the best group and \( N_{wg} \) is the number of individuals in the worst group and “r” is the number of individuals to be shifted between groups, then for the next iteration

\[
N_{bg} = N_{bg} + r
\]

\[
N_{wg} = N_{wg} - r
\]

This process will continue until one or some rules have been completely eliminated by others or the termination criterion has been reached. In this way, the population will automatically select the best dispatching rules while discarding the worthless rules.

4.7. Working of PBHA
For simple problems where process flexibility is not considered, a list of alternate process plans is initially provided. For problems where process flexibility is considered, the date is usually inputted in the form of an AND/OR graph. For the proposed algorithm, the input data are stored in notepad files in the format discussed in Section 3.2. The working of the algorithm is quite simple. A certain number of individuals are initialized. The main purpose of having more than one individual is to prioritize the dispatching rules as already explained in Section 4.6.

An IPPS problem can have more than one optimal solution, so more individuals can help reach alternative solution, instead of just one optimal solution. For the initialized population, the following steps are repeated until the termination criterion has been satisfied. The termination criterion used in the present study is the number of iterations. The flow chart for the algorithm is given in Figure 3.
The first step in problems without process flexibility is the selection of an alternate process plan from the available. This is done randomly. For problems where process flexibility is considered, the first step is to simplify all the AND/OR junctions in every job by randomly selecting a valid combination of operations. Since OR junctions present alternate processing options for a job, this step will also determine the total number of operations required to complete the job, which will be a variable for every iteration.

The next step is the calculation of the probabilities for all the jobs for this iteration. First $P_o$ and $P_c$ are calculated using Equations 3 and 4, respectively, and the final probability for each job is obtained using Equation 4. These job probabilities are then used to select a job at random. If the selected job has more than one chain, then the probabilities for all the chains within that job are calculated using Equation 3. These probabilities are used to select a chain at random. For the selected chain, the first unassigned operation is selected to be assigned to a machine for processing.

The selection of a suitable machine for this operation is done with the help of dispatching rules. Initially, the population is divided into parts, equal to the total number of available dispatching rules. Each dispatching rule is assigned to one part of the population, where it is responsible to select a machine for every operation. So, the operation selected in the previous step is assigned to a machine based on the associated dispatching rule.

Figure 3. Flow chart for PBHA.
These steps are repeated until scheduling for all jobs is completed. In a single iteration, these steps are repeated for all individuals in the population. So, if there are ten individuals in a population, then in a single iteration ten different process plans and schedules will be generated for any given problem.

In the first iteration, the best individual solution is stored and is subsequently used for comparison with the best solution in every iteration. If the best solution in an iteration is better than this stored best solution, it replaces the stored solution. The solutions are also used to adjust the part of population governed by each dispatching rule for the next iteration. Once the termination criteria have been satisfied, the best result is displayed.

5. Experiments & results
The proposed algorithm was coded in C++ and implemented on a computer with Pentium 1.8-GHz dual processor and 1-GB RAM. To validate the efficiency and effectiveness of the algorithm, eight different problems sets were tested. The details of these experiments are given below.

5.1. Experiment 1
The first experiment is one of the simplest and a popular problem, given in Meenakshi Sundaram and Fu (1988), for the evaluation of algorithms for the IPPS problem. The problem is to assign 5 jobs to 3 machines; each job has 4 operations. Although different machines can be used for different operations, there is no flexibility in terms of operation sequencing within the job. The original paper had reported a makespan of 38. This problem has been solved by numerous researchers and the best makespan achieved for this problem is 33 by Lihong and Shengping (2012) among others. When PBHA is applied to this problem, the makespan of 33 is also achieved.

5.2. Experiment 2
This experiment is adopted from Shao et al. (2009) which is a modified version of the original problem presented in Moon and Seo (2005). The problem is to assign 5 jobs with a total of 21 operations to 6 machines. Shao et al. (2009) managed to obtain a makespan of 28 for this problem. Naseri and Ahmed (2012) improved the makespan to 27, the exact solution for this problem, using a hybrid genetic algorithm. Using PBHA, the same makespan of 27 is achieved, but a different plan with a better mean flow time is presented in Table 3 and the corresponding Gantt chart is shown in Figure 4.

5.3. Experiment 3
Experiment 3 consists of data originally presented in Chryssolouris, Pierce, and Dicke (1992) and has been used by Lihong and Shengping (2012), Wong, Leung, Mak and Fung (2006a) and Jain and Elmaraghy (1997), calculating makespan of 5998, 6574 and 6456, respectively. PBHA is able to find an improved makespan of 5388.

<table>
<thead>
<tr>
<th>Table 3. Operation sequencing for Experiment 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>Job1</td>
</tr>
<tr>
<td>Job2</td>
</tr>
<tr>
<td>Job3</td>
</tr>
<tr>
<td>Job4</td>
</tr>
<tr>
<td>Job5</td>
</tr>
</tbody>
</table>

Note: Opr: Operation; M: Selected machine.
5.4. Experiment 4
The data for the fourth experiment are taken from Moon, Lee, Jeong, and Yun (2008); the problem consists of assignment of 13 operations of 5 jobs on 5 different machines. The same problem has been solved by Shao et al. (2009) and Lihong and Shengping (2012) among others. The source paper reported a makespan of 16, while the best achieved makespan is 14, which is also the lower bound for this problem. PBHA is also able to achieve this makespan of 14.

5.5. Experiment 5
The fifth experiment is taken from Lee and Dicesare (1994); it consists of 5 jobs with 4 operations each and 3 machines on which these operations are to be assigned. Lee and Dicesare (1994) had calculated a makespan equal to 439 for this problem, which was improved to 380 by Leung (2010) using ACO. Lihong and Shengping (2012), in a recent paper, have reported a makespan of 360 by applying IGA for this problem, but the best value for this problem was obtained by Li, Zhang, Gao, Li, and Shao (2010) using an agent based approach. When PBHA is applied to this problem, a makespan of 350 is obtained which is the same as the best obtained result, but a different solution is presented. The process plan and Gantt chart are given in Table 4 and Figure 5, respectively.

<table>
<thead>
<tr>
<th>Table 4. Operation sequencing for Experiment 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job1</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Job2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Job3</td>
</tr>
<tr>
<td>Job4</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Job5</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Note: Opr: Operation; M: Selected machine.
5.6. Experiment 6
This experiment is adopted from Lee, Jeong, and Moon (2002) with slight modifications. The original example uses outsourcing of jobs and job due dates which are ignored in this study. The problem consists of assignment of 8 jobs with a total of 20 operations on 5 machines. This problem has also been solved by other researchers like Amin-Naseri and Afshari (2012), Chan et al. (2009), Li et al. (2008). The best reported makespan for this problem is 23 and PBHA is able to reproduce the same result.

5.7. Experiment 7
The problem considered in this experiment is the assignment of 6 jobs with a total of 18 jobs on 5 machines. The problem has been taken from Li, Gao, Shao, Zhang, and Wang (2010) and the makespan of 27 obtained is the best possible solution as reported by Naseri and Ahmed (2012). The same makespan is achieved by PBHA.

5.8. Experiment 8
The problems considered in the experiments thus far did not involve a large number of jobs and machines. The final experiment has been adopted from Kim et al. (2003), which present a comprehensive list of 24 problems based on assignment of up to 300 operations of 18 jobs on 15 machines. The complexity of each job varies providing varying flexibility with respect to routing, sequencing and processing.

These problems have been solved by numerous researchers (Lee, Moon, Bae, & Kim, 2012; Leung, Wong, Mak, & Fung, 2010; Li, Shao, et al., 2010; Lian et al., 2011; Lihong & Shengping, 2012; Wong et al., 2006a; Wong, Leung, Mak, & Fung, 2006b; Zattar, Ferreira, Rodrigues, & de Sousa, 2010) over the past decade in the quest to further improve solution quality and computation time. Lihong and Shengping (2012) have presented the best results among these researches.

5.9. Discussion
A summary of the results for the first seven experiments is presented in Table 5. It is not possible to exactly determine the global minima for an IPPS problem. However, the lower bound for every problem can be calculated and the global minima can never be less than this lower bound. If a solution with a value equal to the lower bound is achieved, then that solution will be a global minimum for the problem. Lower bound, i.e. global minima for six of these problems are already given in the literature. Since heuristic algorithms cannot guarantee global optima, testing these problems helps establishing the effectiveness of the proposed algorithm. For a single-objective IPPS problem, the global optimum may or may not be unique. In two instances, i.e. experiment 2 and experiment 5, different solutions from the one given in the literature have been obtained. Experiment 3 presents a case where the lower bound had not been achieved, but the solution obtained using PBHA presents an improvement on the best solution given in the literature. The results show that PBHA is capable of effectively solving these problems. The computation time for these problems is not reported in the

<table>
<thead>
<tr>
<th>Exp</th>
<th>Original</th>
<th>Best</th>
<th>PBHA</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp1</td>
<td>38 (Meenakshi Sundaram &amp; Fu, 1988)</td>
<td>33* (Lihong &amp; Shengping, 2012)</td>
<td>33</td>
<td>Lower bound*</td>
</tr>
<tr>
<td>Exp2</td>
<td>28 (Shao et al., 2009)</td>
<td>27 (Amin-Naseri &amp; Afshari, 2012)</td>
<td>27</td>
<td>Different solution</td>
</tr>
<tr>
<td>Exp3</td>
<td>6,574 (Leung et al., 2010)</td>
<td>5,998 (Lihong &amp; Shengping, 2012)</td>
<td>5,388</td>
<td>Improved solution</td>
</tr>
<tr>
<td>Exp4</td>
<td>16 (Moon et al., 2008)</td>
<td>14* (Amin-Naseri &amp; Afshari, 2012)</td>
<td>14</td>
<td>Lower bound*</td>
</tr>
<tr>
<td>Exp5</td>
<td>439 (Lee &amp; Dicesare, 1994)</td>
<td>350 (Li, Zhang, et al., 2010)</td>
<td>350</td>
<td>Different solution</td>
</tr>
<tr>
<td>Exp6</td>
<td>23 (Lee et al., 2002)</td>
<td>23* (Amin-Naseri &amp; Afshari, 2012)</td>
<td>23</td>
<td>Lower bound*</td>
</tr>
<tr>
<td>Exp7</td>
<td>27 (Li, Gao, Shao, et al., 2010)</td>
<td>27* (Amin-Naseri &amp; Afshari, 2012)</td>
<td>27</td>
<td>Lower bound*</td>
</tr>
</tbody>
</table>

*Result reported by multiple researchers.
**Lower bound values are taken from Amin-Naseri and Afshari (2012).
Table 6. Comparison of results obtained by PBHA with IGA and SEA

<table>
<thead>
<tr>
<th>Problem</th>
<th>Best makespan value</th>
<th>Average makespan</th>
<th>Computation time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SEA</td>
<td>IGA</td>
<td>PBHA</td>
</tr>
<tr>
<td>1.</td>
<td>428</td>
<td>427</td>
<td>427</td>
</tr>
<tr>
<td>2.</td>
<td>343</td>
<td>343</td>
<td>343</td>
</tr>
<tr>
<td>3.</td>
<td>347</td>
<td>344</td>
<td>344</td>
</tr>
<tr>
<td>4.</td>
<td>306</td>
<td>306</td>
<td>306</td>
</tr>
<tr>
<td>5.</td>
<td>319</td>
<td>304</td>
<td>318</td>
</tr>
<tr>
<td>6.</td>
<td>438</td>
<td>427</td>
<td>427</td>
</tr>
<tr>
<td>7.</td>
<td>372</td>
<td>372</td>
<td>372</td>
</tr>
<tr>
<td>8.</td>
<td>343</td>
<td>342</td>
<td>343</td>
</tr>
<tr>
<td>9.</td>
<td>428</td>
<td>427</td>
<td>427</td>
</tr>
<tr>
<td>10.</td>
<td>443</td>
<td>427</td>
<td>427</td>
</tr>
<tr>
<td>11.</td>
<td>369</td>
<td>368</td>
<td>347</td>
</tr>
<tr>
<td>12.</td>
<td>328</td>
<td>312</td>
<td>318</td>
</tr>
<tr>
<td>13.</td>
<td>452</td>
<td>429</td>
<td>427</td>
</tr>
<tr>
<td>14.</td>
<td>381</td>
<td>386</td>
<td>376</td>
</tr>
<tr>
<td>15.</td>
<td>434</td>
<td>427</td>
<td>427</td>
</tr>
<tr>
<td>16.</td>
<td>454</td>
<td>433</td>
<td>427</td>
</tr>
<tr>
<td>17.</td>
<td>431</td>
<td>415</td>
<td>394</td>
</tr>
<tr>
<td>18.</td>
<td>379</td>
<td>364</td>
<td>352</td>
</tr>
<tr>
<td>19.</td>
<td>490</td>
<td>450</td>
<td>445</td>
</tr>
<tr>
<td>20.</td>
<td>447</td>
<td>429</td>
<td>426</td>
</tr>
<tr>
<td>21.</td>
<td>477</td>
<td>433</td>
<td>427</td>
</tr>
<tr>
<td>22.</td>
<td>534</td>
<td>491</td>
<td>475</td>
</tr>
<tr>
<td>23.</td>
<td>498</td>
<td>465</td>
<td>455</td>
</tr>
<tr>
<td>24.</td>
<td>587</td>
<td>532</td>
<td>526</td>
</tr>
</tbody>
</table>

Note: Lower bound (LB) values are taken from Lihong and Shengping (2012).
*LB achieved.
Figure 7. Gantt chart for experiment 8, problem 16.
Figure 8. Gantt chart for experiment 8, problem 21.
Figure 9. Gantt chart for experiment 8, problem 24.
literature, so it is not possible to do a comparison. The time required to obtain solutions in all seven experiments using PBHA was always less than five seconds.

The 24 problems in experiment 8, all consider process flexibility along with the operation and sequencing flexibility. This added complexity along with the large number of jobs and machines considered in this experiment means that lower bound for only 11 problems has been previously achieved. This problem set has been solved using PBHA and improved results are obtained for 12 of the final 14 problems; these results are shown in Table 6. For 3 of these problems (13, 16 & 21), lower bound has been achieved; the Gantt charts for these problems are given in Figures 6–8, respectively. The Gantt chart for the final problem is given in Figure 9. Owing to its simplicity, PBHA requires a very less computation time to get these results. The code has been executed ten times performing 1000 iterations for each run; the average makespan along with the computation time required are presented in Table 6. If a comparison is made with data given in Lihong and Shengping (2012), it can be seen that compared to IGA better results for all problems, except problem 5, have been achieved in about 20% of the computation time on an inferior computer as shown in Table 6.

6. Conclusions
An algorithm has been presented for the optimization of the IPPS problem. This algorithm incorporates the use of dispatching rules in conjunction with a priority-based assignment system to guide the search process. The algorithm has been tested for a variety of benchmark problems and it can be concluded that the proposed algorithm is capable of producing improved results. Since the working of the algorithm is very simple, the results also show that for larger problems PBHA is significantly faster than other algorithms presented in recent literature.

The present study has been restricted to only evaluation of a single objective function, i.e. the makespan. Minor alterations to the proposed algorithm will enable it to solve multi-objective problems. In the present study, only a handful of dispatching rules were used; a detailed study on the effect different dispatching rules have on the performance of the algorithm can be conducted. The effects of setup and transportation time have also been ignored in this study for the sake of comparison with previous researchers, even though the proposed PBHA is very capable of incorporating the effects of both variable setup and transportation times.

Acknowledgments
The authors thank anonymous referees whose comments helped a lot to improve this paper.

Funding
This research work is supported by National Natural Science Foundation of China [grant number 51035001], [grant number 51421062], [grant number 51375004].

Author details
Muhammad Farhan Ausaf1
E-mail: farhanausaf@gmail.com
Liang Gao1
E-mail: gaoliang@mail.hust.edu.cn
Xinyu Li1
E-mail: lixinyu@mail.hust.edu.cn
Ghiath Al Aqel1
E-mail: gh.alaqel@gmail.com
1State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China.

Citation information
Cite this article as: A priority-based heuristic algorithm (PBHA) for optimizing integrated process planning and scheduling problem, Muhammad Farhan Ausaf, Liang Gao, Xinyu Li & Ghiath Al Aqel, Cogent Engineering (2015), 2: 1070494.

References


Ausaf et al., (2015), 2:1070494

http://dx.doi.org/10.1080/23311916.2015.1070494

Cogent Engineering

http://dx.doi.org/10.1016/j.cie.2007.06.018

http://dx.doi.org/10.1023/A:1008976720878


http://dx.doi.org/10.1080/0951192X.2011.562543

http://dx.doi.org/10.1080/0951192X.2012.684721


http://dx.doi.org/10.1016/j.cor.2008.07.006

http://dx.doi.org/10.1007/s00170-007-1087-8


http://dx.doi.org/10.1023/A:1008952024606


http://dx.doi.org/10.1080/00207540500409723

http://dx.doi.org/10.1080/00207540600675801

http://dx.doi.org/10.1080/00207543.2012.720393

http://dx.doi.org/10.1016/S0360-8352(00)00071-1

http://dx.doi.org/10.1080/0020754981933075

http://dx.doi.org/10.1080/09511921003665775


http://dx.doi.org/10.1080/09511920903207472