

# Real-Time and Online Digital-Print Factory Workflow Optimization\*

Qing Duan, Mukesh Agrawal, Krishnendu Chakrabarty  
Electrical and Computer Engineering, Duke University  
Email: {qd7, mukesh.agrawal, krish}@duke.edu

Jun Zeng, Gary Dispoto  
Hewlett-Packard Laboratories, Hewlett-Packard Co.  
Email: {jun.zeng, gary.dispoto}@hp.com

**Abstract**—On-demand digital-print service offers mass customization and exemplifies personalized manufacturing services. We describe a real-time and online optimization technique based on genetic algorithms (GA) for print factory workflow optimization. We have simulated digital-print factory manufacturing activities as a heterogeneous, concurrent and integrated system. The simulation is based on a virtual print factory, which incorporates real factory characteristics such as successive order acceptances, diverse production lines, various resource types and quantities, and stochastic machine malfunctions. The optimization objective is to reduce the number of orders that miss deadlines, balance resource utilization, and ensure just-in-time production. The optimization technique has been integrated into the virtual factory as a factory scheduler and resource-assignment engine. Significant improvements have been achieved using the GA heuristic compared to baseline methods that are currently implemented in an actual industrial setting.

**Index Terms**—Digital-print factory, workflow optimization, genetic algorithm.

## I. INTRODUCTION

COMMERCIAL print is a highly competitive, customer-centered, and service quality-oriented business with annual retail sales of over US\$700B [1][2]. Establishing a superior service engagement and fulfillment between clients and the print service providers (PSP) is a key to creating value and ensuring profit for a commercial print business [3]. The clients, who are also the content suppliers, request print services and supply contents for print. These services range from traditional publishing agencies for newspapers, books and periodicals to web-based digital media companies, e.g., *www.snapfish.com*. The PSP sells its manufacturing capability to the content suppliers as a form of utility service in exchange for payment. Today, on-demand digital print services is leading the digitization of commercial print and commanding an impressive 8% annual growth [1]. Compared with traditional print services, which produce homogeneous print products (thousands of copies of the same content, also known as “long run-length”), on-demand digital print deals with high-frequency and high-volatility service demands [4].

In the commercial digital print industry, customers can request print products using multiple methods. They may

physically visit the PSP or electronically submit print content to the PSP through the internet. An order is a contract for print service assigned to a PSP. It includes an order identifier, due date, number of copies, price and payment information, customer information, inks and presses, finishing and shipping methods. Several printing processes such as Raster Imaging Process (RIP), cover printing, page printing, folding, collating, binding, packaging and other related processing stages must be scheduled and finished before the completed order can be shipped to customer [5]. Besides offering high quality and good price, the PSP must deliver an order by its deadline since late delivery causes extra shipping cost and reduced customer satisfaction or loyalty.

Physical print artifacts are diverse products that are fabricated through a combination of automated and manual procedures involving heterogeneous equipments. Resources such as equipments, servers and IT infrastructures can malfunction, suffer from anomalies or even break down. Orders (e.g., request for prints) arrive randomly in time and are highly personalized in terms of quantity and quality. All these factors cause the production floor to be subject to high process diversity. Fair resource utilization, Just-In-Time (JIT) production, and job scheduling ensure smooth factory workflow, enhance the factory productivity, and maximize benefits [6]. It has been shown that digital print services are no longer a set of simple rules but a paradigm based on data-driven science where the manufacturing process, resource utilization and work force deployment must be optimized simultaneously to ensure real-time efficiency under complex conditions [7].

In general, continuous order acceptance and job release activities cause the workflow in the factory to be highly variable, the job mix to be dynamic, and efficient resource utilization to be a challenge. There are many factors that must be considered simultaneously for optimization. Orders with short fulfillment lead time have the risk of being delivered late. Urgent and important orders must be given higher priorities. Orders that failed during manufacturing must be rescheduled for reprocessing. Resource utilization should also be balanced by assigning similar amount of workload among the same type of resources to avoid uneven resource overloading and idling. While scheduling orders, the importance of “JIT” and “Opportunity Cost” production management should also be

\* The work of Q. Duan, M. Agrawal, and K. Chakrabarty was supported in part by a grant from HP Labs Open Innovation Research Program.

considered [7] [8] [9]. In this paper, we adopt one of the most popular approaches towards JIT production by modeling it as an earliness-tardiness optimization problem [10]. Therefore, we are able to consider simultaneous scheduling and resource binding in such a successive order acceptance and stochastic order-reprocessing environment. This approach helps to achieve streamlined and efficient production quality in a steady state.

Ad hoc heuristics can no longer manage increasingly complex aspects of print factories; therefore, research on factory workflow optimization is now increasingly important. An example of an outcome of recent research is LPD, Xerox's simulation-based service solution that is intended to boost print shop productivity [11]. To further study this problem, the authors in [1] [12] have defined a digital print ecosystem made up of various entities that have different fulfillment needs and resource requirements. Our virtual factory simulation platform is an extension of this work.

In our recent work [7] and [13] the interrelated scheduling and resource binding problems have been co-optimized. However, the optimization technique has been evaluated for a one-time, static, and fault-free factory environment, which is an unrealistic case of the heterogeneous environment in the virtual factory.

In this paper, a GA-based heuristic optimization technique is presented to perform simultaneous scheduling and resource binding. This technique has been successfully integrated into the virtual factory simulation platform. The results are obtained after the factory has been simulated for seven days, i.e. the time elapsed in the factory.

The main contributions of this paper are listed below:

- The digital-print factory workflow optimization problem is formulated under due-date constraints, resource contentions, and the Just-In-Time production scenario. The proposed GA-based heuristic provides simultaneous task scheduling and resource binding solutions;
- A heterogeneous virtual print factory simulator has been developed to mimic the activities, characteristics, and functionalities of a real print factory. The GA-based optimization technique has been integrated into the virtual print factory to realize real-time, online optimization in a successive order acceptance and stochastic order reprocessing environment;
- The GA heuristic takes both new and failed orders reprocessing into consideration while performing optimization;
- A bottom-up scheduling scheme has been designed to improve JIT production service quality;
- In order to achieve online optimization, critical features of the print factory such as resource status are continuously monitored and updated. By combining these features with the performance of GA, human managers can identify bottleneck processing stages in the factory and make strategic decisions such as upgrading or purchasing resources.

The rest of the paper is organized as follows. In Section II, we introduce digital-print manufacturing operations and describe the discrete-event simulation platform. Section III describes the problem formulation and modeling. Section IV

and section V present our GA heuristic and the baseline methods currently in use, respectively. Simulation results are presented in Section VI. Conclusions and future research directions are in Section VII.

## II. DISCRETE-EVENT SIMULATION OF PRINT FACTORY

It is very challenging to achieve high productivity in digital printing with non-homogeneous arrival of orders and varying customer requirements [1] [11]. The Ptolemy based simulation platform was first developed in [1] for simulating commercial digital print operations. Its primary utilization was for understanding various lean manufacturing rules and runtime policies [14]. This platform is further used to evaluate optimization methods and decisions before their adoption on the real factory floor. The idea of modeling document production as a manufacturing process was also described in [11]. Productivity solutions based on concepts derived from operations research were developed and deployed at several sites to accomplish substantial productivity and cost improvements for print shops [11].

The demand-driven model of "Just-In-Time" fulfillment motivates digital PSPs to emphasize innovations in lean manufacturing productivity maximization [15] [16]. Reischling Press, Inc. (RPI) is an example of such a PSP whose fulfillment procedures are centered on the concept of the "value stream" [17]. Figure 1 shows the RPI book factory organization. The first part is IT services shared by all value streams, including order admission, production planning, pre-press and job release. Following the pre-press, a scheduler application evaluates order priority, decomposes orders into products, and dispatches products towards different fulfillment paths. Cover making and book block printing are the following two main streamlines. After all the products of an order have arrived at the shipping buffer, this order is assembled, packaged and shipped. The end-to-end workflow is monitored at each product level by the audit function performed by either software or wireless barcode scanners via operators. Once a part of an order fails, it is sent back to the pre-press stage for reprocessing. Every processing stage may fail following a certain stochastic probability distribution that can be derived from history data.

Figure 2 illustrates the architecture of the operations simulation software [18] developed as a virtual factory simulation platform based on Ptolemy [19]. The order stream is acquired from the printers' enterprise resource management (ERM) system, which encompasses both the enterprise resource planning (ERP) system and the manufacturing execution system (MES) [12]; the fulfillment paths, resources and the operating policies are programmed in the model. MySQL, another open-source code, has been integrated as the database manager for all the input data and simulation results. Visual analytics tools interface with MySQL to display or extract both the factory level performance and cell-machine level dynamics. They also provide automated query and correlation discovery among different attributes.

The end-to-end fulfillment processes implemented and order data utilized in the virtual factory were directly pulled from

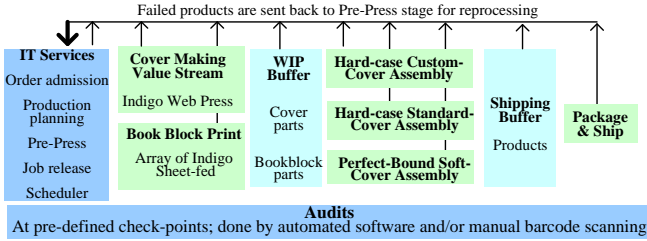


Fig. 1: An example of a lean manufacturing process for digital printing (Source: RPI).

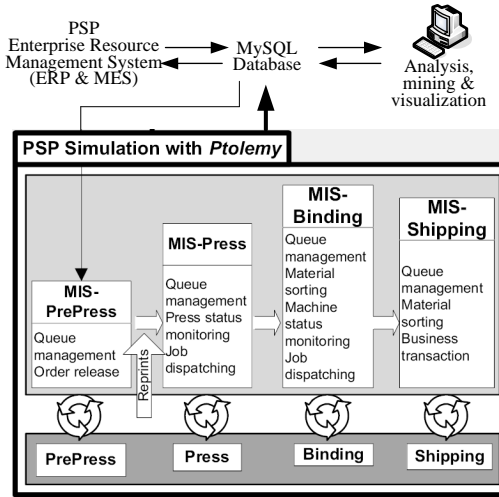


Fig. 2: “Virtual Print Factory” simulation platform architecture.

RPI’s enterprise resource planning database, and integrated into the virtual factory. The results generated by the simulation platform with RPI’s internal factory audit data were validated and found to shown good correlation with the situation on the real factory floor [12].

The benefit of this simulation platform is that it provides an environment that imitates the real situation on the print factory floor. Hence, new optimization techniques as a new scheduler application can be evaluated, as shown in Figure 1.

### III. PROBLEM DESCRIPTION AND FORMULATION

In this section, the main components, behaviors and critical properties in a print factory are described. Following this, we show how the optimization problem is formulated and simulated using the virtual factory simulation platform.

#### A. Resources

The fulfillment of an order involves the processing of a sequence of steps that are serviced by different types of resources. Each resource is characterized by its functionality and speed. In the virtual factory, each resource is implemented by a functional block, which is specific to this type of resource [12]. The processing time for every resource follows

a Gaussian distribution with a mean and a standard deviation. Resource reliability is simulated by a stochastic malfunction mechanism as well. All speed and malfunction data must be carefully extracted from real factory data to emulate the factory manufacturing process. Resource malfunction must be integrated into the virtual factory since other than tight order deadlines or large orders, the randomness introduced by stochastic resource malfunction is another major cause of late delivery.

#### B. Order, product and part

The payload of an order is composed of several book titles. Each book title is referred to as a *product*. A product’s information contains the digital file that will be used for preprocessing and printing, quantity (number of copies) and a fulfillment path ID. There are several types of products a factory can offer. Every type of product corresponds to its own *fulfillment path*, which defines a sequence of tasks or processing steps that completes the creation of a product of this type. A product may be further composed of more than one *part*. In case of multi-part products, the corresponding fulfillment path starts with parallel fabrication of the parts (e.g., printing the book cover and printing the book block), followed by part assembly into a product. After all products are fabricated, they can be merged to form a complete order. Based on these concepts, a processing stage can be categorized into an order-level, a product-level or a part-level stage [7].

Figure 3 shows an example of two simplified fulfillment paths. These paths were used in our prior research for evaluating the optimization method in [7] [13] as test cases. The fulfillment paths simulated in the virtual factory correspond to the streamlines in the real print factory and are considerably more complicated and lengthy. In Figure 3, there are two different products within one order and they have fulfillment path A and fulfillment path B, respectively. Fulfillment path A has two parts, one part is used for printing cover and the second part for printing a book block. These two parts join together at the *Binding* stage to indicate the end of part processing. These two parts can be processed in parallel and are independent of each other, while *Binding* stage requires both parts to be completed before the *Binding* task of a product begins. Binding is therefore a product-level processing stage. The fulfillment path B processes a single-part product. The order-level stages *Serializing* and *Packaging* are common to all products.

#### C. Successive order acceptance

In a real factory, orders are accepted in a dynamic and heterogeneous manner. The virtual factory follows exactly the same order acceptance pattern. The state of a virtual factory under a normal incoming order frequency and quantity is illustrated in Figure 4. The horizontal axis shows the factory time, the vertical axis shows the number of orders accepted at 15-minute intervals.

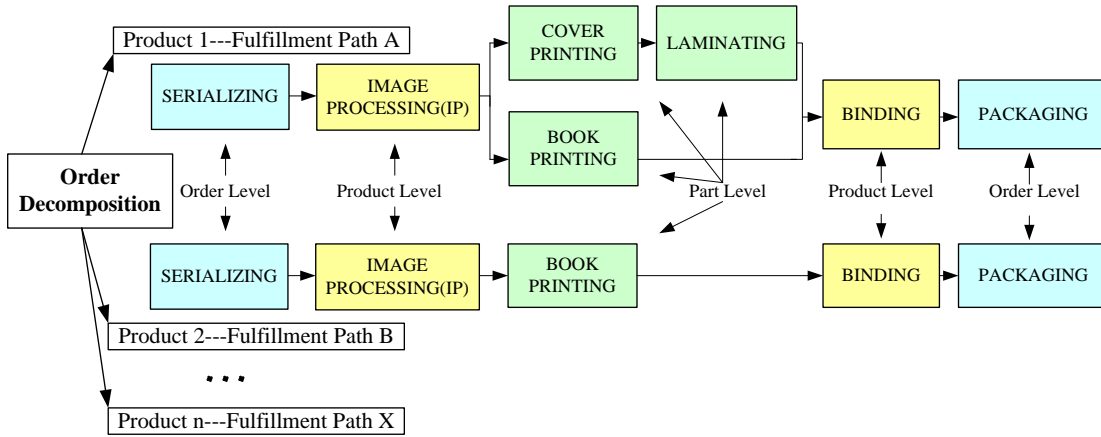


Fig. 3: An illustration of fulfillment paths.

#### D. Stochastic product reprocessing

All resources are subject to stochastic malfunctions caused by various failure modes on the factory floor. It results in reworking of some products. Due to the unrecoverable nature of physical print artifacts, no matter at which processing stage a product fails, this product is sent back to the pre-press stage in order to be reprocessed from the beginning as shown in Figure 1. These activities are implemented in the virtual factory as well, so that, once a product fails at any processing stage, its current manufacturing process is aborted and it is sent back to the pre-press stage immediately. If one or more products in a multi-product order require reworking, all other good products have to be stored and wait until all the reprocessed products are ready to be assembled into one order.

#### E. Task sequencing graph

It has been reported recently that the digital print automation problem has striking similarities with electronic design automation (EDA) especially in high-level synthesis [7][20][21]. The sequencing problem in high-level synthesis is modeled using a sequencing graph that captures dependencies between operations. The problems of scheduling and resource binding are solved using heuristic methods that are scalable for large designs. The digital-print optimization problem can be modeled using high-level synthesis; state-of-the-art EDA algorithms can be leveraged to solve this problem [20].

The analogy can be illustrated by the problem of simultaneous assignment of print operations or tasks to time steps (scheduling) and selection of resources for these tasks (binding). In this way, it can be shown that how methods established in the design automation arena is adapted for optimization problems in digital printing. We can model print jobs, the relationships between them, and dependencies between tasks within a job, in terms of task sequencing graphs (TSG) as shown in Figure 5 [20]. This formal representation can then be used for scheduling and resource binding. A procedure to obtain the task sequencing graph for all print operations at the system level is discussed in [7]. Therefore, the precedence constraints and resource contentions among print tasks are

captured by *sequencing edges* and *resource contention edges* respectively.

#### F. Problem formulation

The optimization problem is now formulated as follows. A detailed description of all the available resources on the factory floor is known and stored in the resource management database. Variable  $l$  denotes the index of an order. The virtual factory starts running and accepting order  $l$  with deadline  $D_l$ . Once accepted, order  $l$  is divided into products. We attempt to create a priority sequence among all the products that belong to either newly accepted orders or reprocessed orders. While the factory keeps running, orders can be completed and the actual completion time  $C_l$  for order  $l$  is obtained. The difference  $S_l$  between the deadline  $D_l$  and the completion time  $C_l$  for order  $l$  is called *slack time*. The slack time is negative if an order misses its deadline.

Our primary objective is to reduce the number of late orders (i.e. orders that miss their deadlines), which can be expressed in terms of the metric “Missed Order Ratio” ( $MOR$ ). Suppose there are  $N_1$  orders that must be completed by a certain deadline; however, simulation shows that  $N_2$  orders ( $N_2 \leq N_1$ ) are completed after this deadline. Thus, the  $MOR$  can be calculated as follows:

$$MOR = \frac{N_2}{N_1} \times 100\% \quad (1)$$

Our secondary objective is to achieve a high quality of JIT production [12]. There is an *earliness penalty*  $e$  for order completed earlier than its deadline and a *tardiness penalty*  $t$  for order completed after its deadline. For every order, the earliest possible completion time  $EC$  is estimated through averaged history data. In order to achieve high quality of JIT production, the JIT penalty which is denoted by  $P_{JIT}$  has to be reduced. Suppose that there are  $N$  orders, parameter  $P_{JIT}$  can be estimated as follows:

$$P_{JIT} = \sum_{i=1}^N \left\{ e_i \cdot \frac{1}{(D_i - EC_i)} \cdot \max(D_i - C_i, 0) + t_i \cdot \frac{(C_i - EC_i)}{(D_i - EC_i)(C_i - D_i)} \cdot \max(C_i - D_i, 0) \right\} \quad (2)$$

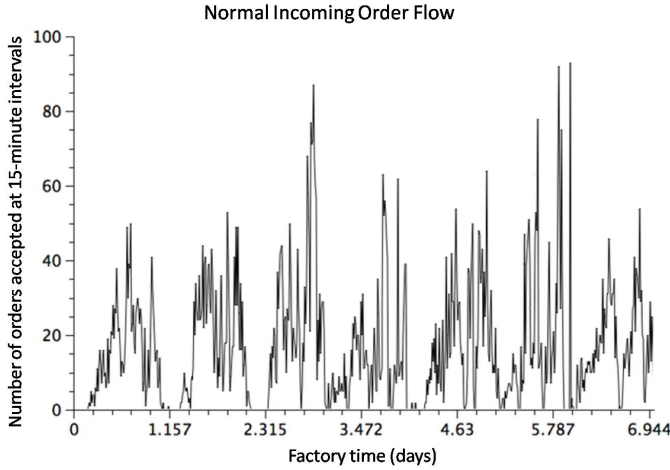


Fig. 4: An example of normal incoming order flow.

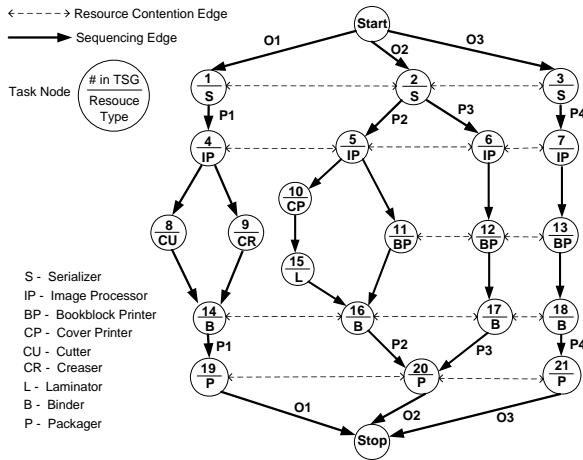


Fig. 5: An example of a task sequencing graph.

In comparison, the Opportunity Cost which is represented by parameter  $OC$  is considered only for orders that completed ahead of their deadlines [8]; therefore:

$$OC = \sum_{l=1}^N \left\{ e_l \cdot \frac{1}{(D_l - EC_l)} \cdot \max(D_l - C_l, 0) \right\} \quad (3)$$

For simplicity, we assume that  $oc$ ,  $e$  and  $t$  are tunable parameters, depending on how much we want to penalize each scenario. For example,  $t_l$  can be set much higher than  $e_l$  in order to emphasize the loss caused by a late order  $l$ . In a real factory, this choice may vary depend on each order's attributes, such as order importance, profit, and size.

Even for a single machine, the problem of managing workflows in terms of sequencing with earliness and tardiness penalties is an NP-complete problem [20] [22]. If only one processing stage is considered, our problem is equivalent to the single-machine problem with deadline penalty. Therefore, the general problem that we are dealing with is computationally at least as hard as known NP-complete problems. Good heuristic

methods are therefore needed for practical problem instances.

In this work, an advanced optimization technique is integrated into the virtual factory simulation platform to serve as a product scheduler and resource assignment indicator application. The robustness of this new technique under successive order acceptance and stochastic order failure is evaluated in the virtual factory, which mimics the real factory floor. The detailed explanation of this GA-based heuristics optimization technique is presented in the following section.

#### IV. GA-BASED HEURISTIC ALGORITHMS

GA-based strategies have been recognized as efficient solutions for solving optimization and search problems across various domains [23]. In our problem, we attempt to optimize resource binding while satisfying due-date commitments and JIT production requirements. By maintaining a pool of candidates and attempting to improve the population in each generation using evolutionary techniques, GAs can effectively obtain near-optimal solutions for such constrained optimization problems. This section extends the work presented in [7], and describes major enhancements to accommodate real-world complex situations in an actual print factory.

##### A. Order set, resource set and task set

An order set contains all newly accepted orders and all failed products that have been sent back to the pre-press stage during each 15-minute interval in the virtual factory. The virtual factory is set to continuously run for seven days on the simulation platform; therefore, 672 (i.e. 7 days/15 minutes) diverse order sets have been generated. After accumulating all new orders and failed products into one order set, a TSG is constructed for this order set. A failed product is treated as a single-product order in the TSG. The accepted time of this failed product is updated based on the time stamp when it is sent back to the pre-press stage and its deadline is inherited from its parent order. The available processing duration for this failed product is therefore reduced.

A *resource set* refers to the resources that have the same functionality. On one hand, resources belonging to the same resource set can process similar tasks. On the other hand, they can have different capabilities, e.g. speed and reliability.

A *task set* is defined as a set consisting of similar tasks that are competing with each other for the resources within the same resource set. There is a one-to-one correspondence between task sets and resource sets. In Figure 5, a task set and its corresponding resource set are connected by resource contention lines. Task sets are updated with the order set every 15 minutes. The resource set information is also refreshed if the resource status in the factory has changed. For example, if a resource breaks down, this information is immediately updated in the resource management database. This resource is temporarily removed from the resource set list until it is recovered.

##### B. Average waiting time for a resource

Tasks assigned to the same resource cannot be processed simultaneously. Variables  $i$  and  $k$  denote the index of a task

and a resource, respectively. If a task  $v_i$  arrives at a resource  $r_k$  when  $r_k$  is busy with its current job, then task  $v_i$  has to wait in the queue of  $r_k$  to be served. The earliest possible start time for a task to be processed is its arrival time at its assigned resource. It also equals the latest completion time of all its preceding tasks. Therefore, we define average waiting time for a resource  $AWT(r_k)$  to be the average duration a task needs to wait in the queue of  $r_k$  to get processed. Function  $AWT$  is a time-varying function that depends on the resource queue length. If many tasks assigned to the same resource, its  $AWT$  increases. The value of  $AWT$  is dynamically obtained from the virtual factory simulation and refreshed as input to the GA every 15 minutes to guarantee that the resource status is updated to reflect the up-to-the-moment situation. Parameter  $AWT$  is important for estimating the general-case task execution duration  $w(v_i)$ , defined as follows:

$$w(v_i) = AWT(r_k) + t(r_k, v_i) \quad (4)$$

where  $t(r_k, v_i)$  is the processing time of task  $v_i$  if processed by resource  $r_k$ . In contrast to [7], we do not consider the overly conservative worst-case estimation of task execution duration.

### C. Product-level priority assignment

Priority assignment enable the factory to manage the execution sequence of tasks that are competing for the same set of resources. In [7] and [13], we developed a task-level scheduling and resource binding scheme that assigns a specific processing schedule and resource allocation for every task in the TSG. As shown in Figure 6a, tasks from different products can be assigned to the same resource and they are connected by resource contention lines. Resources perform tasks according to task priority. However, on the real factory floor, task-level scheduling is impractical and cannot be achieved in real-time. Instead of retaining and rearranging tasks to achieve their priority sequence and process them according to predefined time slots, resources follow the rule of “first-come, first-served” to avoid wasting free time slots. Resources process tasks in the sequence in which they arrive. Tasks are retained until it is their turn to be processed.

Therefore, we adopt product-level priority assignment. This strategy assigns priorities to products rather than tasks. Tasks in one product have the same priority. The first tasks of products are released according to product-level priority. The remaining tasks’ execution sequence can usually be maintained since resources do not rearrange tasks, as shown in Figure 6b. Each failed product is removed from its original manufacturing flow and immediately sent back to the pre-press stage. It is added to a new order set, so that it can obtain new priority through another run of GA. Task-level resource binding is still practical by sending a task to its pre-assigned resource. Therefore, product-level priority assignment and task-level resource binding strategy are combined. Hence, GA computes the optimal priorities for products and the resource assignments for tasks. This strategy is compatible with the situation in a real factory.

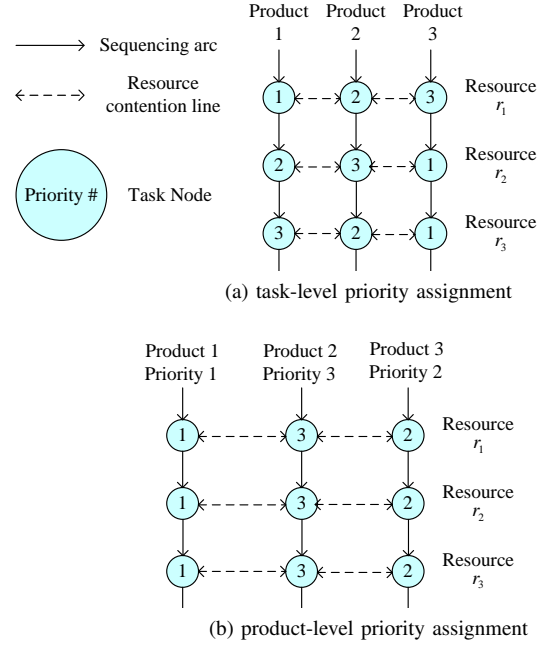


Fig. 6: Priority assignment strategies.

### D. Chromosome representation and decoding

Scheduling and binding solution can be represented as a *Chromosome*, which is a vector of *genes*. A random key representation technique is utilized, which allow us to realize a good chromosome representation for easy and unambiguous decoding of a chromosome into a valid solution [24]. Each gene in the chromosome is a random fractional number between 0 and 1. The first half of the genes is used for encoding the priorities of the tasks and the next half of the genes is used for allocating resources to all the tasks. Note that tasks belonging to the same product have the same gene value since they inherit their parent product’s priority. Another advantage of such chromosome representation is that after applying simple evolutionary techniques, such as crossover the resulting chromosome is still a valid solution. The detailed explanation of how to decode a chromosome can be found in [7].

### E. Bottom-up scheduling

Instead of initiating scheduling from the top of the TSG (i.e., first tasks of orders in one order set [7] [13]) as the manufacturing flow does, the new scheduling scheme starts from the bottom of the TSG (i.e. the last tasks of orders). GAs that adopt this scheduling strategy are referred to as bottom-up GAs. There are two primary reasons for deploying bottom-up scheduling. First, it naturally optimizes order-level JIT production rule. If the GA sets the last tasks to be completed as close as possible to their deadlines depending on the availability of resources, orders are scheduled to be accomplished right before their deadlines. Second, rather than scheduling orders from the same reference timestamp in a top-down manner, the bottom-up GA computes different start times for all the products. Start time determines product priority. Therefore, according to start time, a sequence for

dispatching all the products can be obtained. On the real factory floor, it is hard to ensure that products are dispatched at predefined timestamps. For example, a product's predefined dispatching timestamp has already passed before the product is ready to be dispatched for manufacturing. However, it is easy to dispatch them following a certain sequence.

Figure 7 outlines the heuristic procedure based on bottom-up GA. Each task is assigned a deadline by propagating the order deadlines upwards in the TSG. For the last task of each order, its deadline is the same as the order deadline. Here, variables  $i$  and  $j$  denote the index of a task. The deadline  $\delta(v_i)$  for task  $v_i$  is computed from deadlines of its child tasks using the following equation:

$$\delta(v_i) = \min_{v_j} \{\delta(v_j) - \omega(v_j) | v_j \in \text{childs}(v_i)\} \quad (5)$$

where the execution duration  $\omega(v_j)$  is obtained by applying equation (4). The parameter *start time* of task  $v_i$  represented by  $st_i$  is the time when it starts being executed by its assigned resource. Similarly, *end time* represented by  $en_i$  is the time when it is completed. If two tasks are assigned to the same resource, the conflict between them is resolved by using the priority values inherited from their parent products. A task can only be scheduled if: (i) its child tasks have been scheduled; and (ii) the resource assigned to it is free to use. Note that high priority products are assigned with earlier start times than low priority products. In bottom-up GA, however, low priority products are scheduled first followed by high priority products.

#### F. Fitness function

From each feasible scheduling and binding solution, every task obtains a start time and an end time. It is possible that, a task is ready to be processed long before its computed start time. This situation happens when its assigned resource ought to process other tasks first, so that this task cannot be processed as soon as it is ready. We define *waiting time*  $wt_i$  for task  $v_i$  as follows:

$$wt_i = \max_{v_j} \{en_j | v_j \in \text{parents}(v_i)\} - st_i \quad (6)$$

Therefore, parameter  $wt_i$  is the duration that task  $v_i$  has to wait before being processed. The objective function is to minimize the sum of all tasks' waiting time. In GA a chromosome that is most likely to become a solution should have a higher fitness value than other candidates. A schedule with a lower value of the objective function is regarded as better than all other schedules with higher values. Suppose there are a total of  $M$  tasks in a TSG. Therefore, the fitness function is defined as the negative of the objective function and it is calculated as follows:

$$f = -\left(\sum_{i=1}^M wt_i\right) \quad (7)$$

It is observed that this fitness function helps to achieve the resource-binding objective of balancing the resource utilization. Utilization is measured in terms of time. It is the total time that a resource is executing tasks. Notation  $\mathcal{R}$  is the set of all the resources in the factory. Variable  $n_{\mathcal{R}}$  denotes the number of different types of resources. Variable  $n$  denotes the resource type,  $1 \leq n \leq n_{\mathcal{R}}$ . Notation  $R_n$  denotes the set of resources of type  $n$ . Fair resource utilization is achieved

when in each resource set the utilization of the most heavily loaded resource is close to the utilization duration of the least loaded resource. The utilization of resource  $r_k$  is calculated as follows:

$$u(r_k) = \sum_{i=1}^M t(r_k, v_i) \cdot \delta_{ki} \quad (8)$$

where  $t(r_k, v_i)$  is the processing time of task  $v_i$  if processed by resource  $r_k$ . Variable  $\delta_{ki}$  is a binary indicator, it equals 1 if task  $v_i$  is scheduled to be processed by resource  $r_k$ , if resource  $r_k$  is not assigned to task  $v_i$  then  $\delta_{ki}$  equals zero.

The objective is to minimize the sum of the normalized utilizations of the most heavily loaded resources from all the resource sets. A secondary fitness function is defined as the negative value of this objective:

$$f_2 = -\left(\sum_{i=1}^{i_{\mathcal{R}}} \frac{\max\{u(r_k) | r_k \in R_i\}}{\sum_{r_k \in R_i} u(r_k)}\right) \quad (9)$$

If the secondary fitness function is set as the fitness function in GA, the computation converges very fast. Therefore, a two-step method is used. First the GA obtains the best set of chromosomes with the same maximum fitness value computed by the first fitness function  $f$ . The GA then selects the one with the highest fitness value computed by the secondary fitness function  $f_2$  from the best chromosome set.

It is possible that a task is scheduled even earlier than the accepted time of its parent order through bottom-up scheduling. This situation occurs when an order deadline is very tight. However, this situation happens rarely on a real factory floor, since in general, most deadlines are reasonably assigned. Furthermore, an invalid start time shows a task is already behind schedule. Therefore, an invalid start time means high priority for this product. Products with invalid start times are usually immediately dispatched.

Note also that there is an important difference between  $C_l$  and  $en_i$ . The value of  $C_l$  is the completion time of the order  $l$  obtained by running the virtual factory on the simulation platform. The value of  $en_i$  is the completion time of a task  $v_i$  obtained by scheduling according to a chromosome in GA. Lower-case notations are used to denote the values obtained by GA; upper-case notations are used to denote the values obtained through running the virtual factory as well as constant values such as order deadlines.

#### G. Evolution strategies and computational complexity

Three evolutionary approaches, namely, *reproduction*, *crossover* and *mutation* are used for generating a new generation of candidates. *Reproduction* is defined as the copying of top chromosomes from one generation to the next [24]. This step ensures the preservation of candidates that are at least as good as what are present in the current population. The top 30% of the population undergoes reproduction in this heuristic algorithm. *Crossover* enables the exchange of genes between two chromosomes. The parameterized uniform crossover approach is used in this work [25]. The crossover step accounts for 50% of the total population in the next generation. While performing crossover, tasks within the same product should maintain the same gene value. *Mutation* is

achieved by randomly creating chromosomes for the remaining portion of the next-generation population. The evolutionary approaches are the same as in our recent research [7] [13]. The computational complexity analysis is similar to a top-down GA heuristic algorithm [7]. However, a reduced computational complexity is achieved since the number of products is considerably less than the number of tasks.

#### H. GA Integration into the virtual print factory

The integration of GA into the virtual print factory is shown in Figure 11. Under the successive order acceptance and stochastic order failure environment implemented in the virtual factory, GA is applied every 15 minutes for every order set. When GA finishes computation, all tasks get start times and resource assignments. A product's start time equals the start time of its first task. While GA is running, these products are being pre-processed in parallel through the stages only involving software, which means no physical copies have been manufactured yet. Therefore, before products reach the first physical-copy processing stage, GA terminates and their start times are already available. The maximum duration for running GA once is set to 15 minutes because the pre-press stage takes at least 15 minutes to pre-process a product.

A new stage, called the dispatching stage, is created and placed before all physical-copy stages. As mentioned before, it is unreasonable to dispatch products precisely at a certain timestamp. There are two main reasons. First, we do not want to overload the following physical-copy stages by continuously sending tasks to them. Instead, the dispatching stage maintains a product pool and arranges all these products by start time. Start time serves as priority parameter for products. When there is a resource available, a dispatching signal will be sent to the dispatching stage. The product with the earliest start time will be sent to that resource since its start time means it has the highest priority. Second, start times may be even earlier than the real time when products reach the dispatching stage. If one product has an invalid start time earlier than current time that probably means its parent order is approaching its deadline. This product can be dispatched very soon for manufacturing to catch up with its parent order's deadline. In contrast, if a product has a rather late start time, then this product can be maintained in the dispatching stage for a while. By dispatching products according to start time, the scheduler attempts to realize JIT production in a realistic manner.

#### V. RULE-BASED BASELINE METHOD

The rule-based baseline methods are mainly EDD (earliest-due-date) and Min-Slack (minimum-slack-time) heuristics. The EDD heuristic implements order-level priority scheduling. Orders are ranked by their deadlines and all products from the same order have the same priority. The Min-Slack heuristic implements product-level priority scheduling. Each product inherits its parent order's deadline. By considering slack time (i.e. the difference between the deadline and the estimated completion time) for each product, Min-Slack heuristic is more robust than EDD heuristic which considers deadline as the only factor while assigning priority. Min-Slack heuristic can

```

1: Initialize all the resources' latest free time as infinity
2: for each task  $v_i$  in the system do
3:   Compute  $\omega(v_i)$ , general-case execution duration
4:   Initialize  $\delta(v_i)$ , deadline of every task  $v_i$ 
5: end for
6: Generate Initial Population  $P(t)$  of chromosomes
7: while termination criterion not reached do
8:   for each chromosome of  $P(t)$  do
9:     Create schedule by the procedure in Figure 8
10:    Compute  $wt_i$ , waiting time of every task  $v_i$ 
11:    Determine fitness value,  $f$ 
12:   end for
13:   Perform evolutionary steps: reproduction, cross over,
      mutation, obtain  $P(t+1)$ ,  $P(t) \leftarrow P(t+1)$ 
14: end while

```

Fig. 7: Bottom-up GA heuristic procedure.

```

1: for each task set  $T$  do
2:   for each task  $v_i \in T$  do
3:     Assign resource and priority by decoding a chromosome
4:   end for
5:   Sort  $T$  based on deadline  $\delta(v_i)$  in descending order
6:   Sort  $T$  based on priority  $\rho(v_i)$ 
7:   Schedule all tasks in  $T$  by the procedure in Figure 9
8: end for

```

Fig. 8: Procedure to create a schedule for a task set.

be summarized as shown in Figure 10. Function  $delay(p_i)$  returns a duration and varies linearly with the product's slack time. Function  $deadline(p_i)$  returns the last tasks's deadline of product  $p_i$  obtained by equation (5). The less the slack time, the smaller the delay. The resource assignment strategy carried out in Min-Slack heuristic is simply round-robin. This means that resources in the same resource set take turns to execute tasks.

Min-Slack is superior to EDD in the sense that it reasonably schedules orders of different sizes but with the same deadline by considering slack time. EDD omits the importance of the available processing duration.

### VI. RESULTS AND COMPARISON

#### A. Simulation settings

In a real factory, no resource is 100% reliable. The stochastic nature of resource malfunction is simulated by a probability distribution function in the virtual factory. Whenever a resource processes a task, there is a failure probability that this resource fails to process it. The more reliable the resource is, the lower the probability of failure. In order to compare the GA heuristic with the Min-Slack heuristic, the same incoming order flows and the same resources are used in both cases. However, the same resource malfunction events can never be repeated due to such stochastic nature of resource malfunction. On the other hand, the ratio of the amount of reprocessed products over the total amount of products is nearly equal in both cases. Even though it is impossible to predict at which processing stage a product will fail, the ratio of the number of failed products over the total products at any processing stage is almost constant since the total number of products is large.

Several comparison simulations were conducted under four different situations and always obtained similar results if the same factory settings were adopted. Therefore, we randomly selected one result from each situation and present them here.

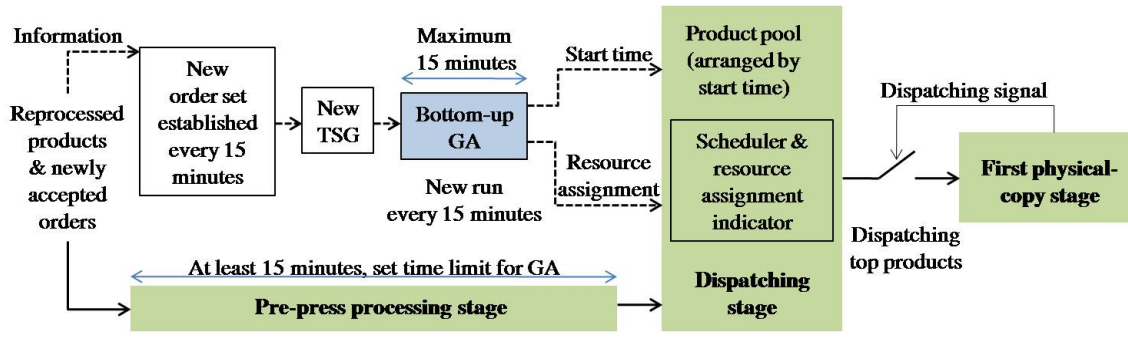


Fig. 11: Illustration of GA integration into the virtual print factory.

TABLE I: Comparison results under  $\times 1.0$  factory workload (JIT penalty is normalized).

Order group: number of orders   same deadline	MOR (GA Heuristics)	MOR (Min-Slack Heuristics)	JIT penalty (GA Heuristics)	JIT penalty (Min-Slack Heuristics)
Group1: 381 322200	0%	7.10%	0.00078044	0.05809712
Group2: 778 408600	0%	8.90 %	0.00153737	0.13784141
Group3: 1114 495000	0.45%	26.10 %	0.00671823	0.57924994
Group4: 1535 581400	2.10%	31.00 %	0.00669258	1.0

```

1: for task  $v_i$  do
2:   Set  $end_1$  as assigned resource  $k$ 's latest free time
3:   if task  $v_i$  has no child task then
4:     Set  $end_2$  equals parent order deadline
5:   else
6:     Set
7:   end if
8:   Update  $en_i = \min \{end_1, end_2\}$ 
9:   Update  $st_i = en_i - t(r_k, v_i)$ 
10:  Update resource  $k$ 's latest free time equals  $st_i$ 
11: end for

```

Fig. 9: Procedure to create a schedule for a task.

All the simulations were set to be run for seven days in terms of the time elapsed in the factory. We compared the performance of new optimization techniques with the rule-based heuristics under normal incoming order flow, dense order flow, even denser order flow, and normal order flow with tight order deadlines.

For simplicity, the earliness penalty  $e$  and opportunity cost parameter  $oc$  were set to 1 and the lateness penalty  $t$  was set to 20 for all the orders. The factory time resolution was simulated in seconds. Order deadlines were determined by a set of shipping times in the virtual factory. Although orders were accepted randomly during the seven days as shown in Figure 4, the amount of available distinct shipping times were much less because orders were shipped out together in batches rather than shipped out as soon as an order was completed.

Therefore, orders can be divided into groups on the basis of deadline. For example, in Table I, the first column shows that there are four groups of orders. In the first group, 381 orders have the same deadline of 322,200 seconds (approximately 3.7 days, factory time starts at 0 seconds), and in the second group there are 778 orders with the same deadline of 408,600 seconds (approximately 4.7 days). We only look at orders that are accepted after 200,000 seconds (approximately 2.3 days) and due before 604,800 seconds (7 days). It is observed that after 200,000 seconds, the virtual factory reached a stable full-

```

1: for each incoming and reprocessing product  $p_i$  do
2:   Estimate processing duration as  $cost(p_i)$ 
3:   Order priority  $\rho_o(p_i)$  equals order deadline this product belongs to
4:   Shop priority  $\rho_s(p_i)$  equals accepted time  $A_{p_i} + cost(p_i)$ 
5:   if  $\rho_s(p_i) > \rho_o(p_i)$  then
6:     Late priority  $\rho_l(p_i) = 0$ 
7:   else
8:     Late priority  $\rho_l(p_i) = 1$ 
9:   end if
10:  if  $p_i$  fulfillment path includes CoverPrinting then
11:    Value Stream priority  $\rho_v(p_i) = 0$ 
12:  else
13:    Value Stream priority  $\rho_v(p_i) = 1$ 
14:  end if
15: end for
16: Sort all the products arrived in the last 15 minutes by:
    $\rho_l(p_i), \rho_s(p_i), \rho_v(p_i)$  and obtain a new priority sequence
17: for each product  $p_i$  in this new order do
18:   Calculate  $slack(p_i) = deadline(p_i) - currenttime$ 
19:   if  $slacktime(p_i) < 0$  then
20:     Distribute this product at once
21:   else
22:     Distribute this product after certain delay  $delay(p_i)$ 
23:   end if
24: end for

```

Fig. 10: Rule-based Min-Slack heuristics.

utilized state when all the resources were occupied most of the time. This state is detected by nonzero queue length for almost all the resources. This implies the onset of the factory's full utilization. The JIT penalty was computed and normalized to the maximum penalty value occurred in each simulation. The normalized JIT penalty value ranged from 0 to 1. A value of 1 indicates the highest JIT penalty and therefore, the worst JIT production realization.

## B. Simulation results

Under each of the four representative situations, GA and the Min-Slack heuristics were compared. The comparison results

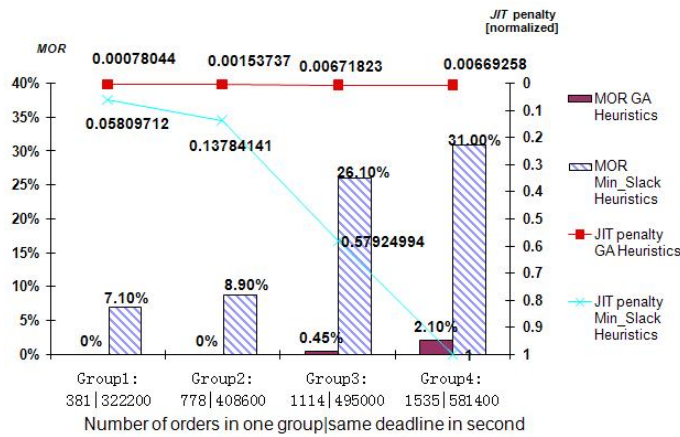


Fig. 12: *MOR* and normalized JIT penalty of order groups under  $\times 1.0$  factory workload.

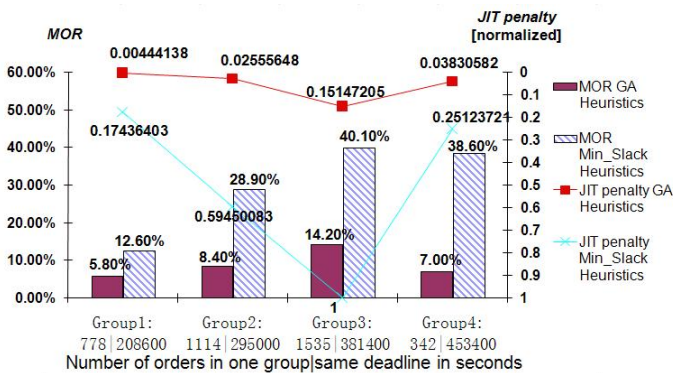


Fig. 13: *MOR* and normalized JIT penalty of order groups under  $\times 1.0$  factory workload and tighter deadlines.

are presented as follows.

**Situation 1:**  $\times 1.0$  order flow, normal deadlines.

The simulation was ran under normal ( $\times 1.0$ ) workload; the results are shown in Figure 12. As mentioned before, these are real order data obtained from a commercial PSP. The simulation results are shown in Table I and Figure 12 in two different ways. For example, the first row in Table I shows that there were 381 orders with a deadline of 322,220 seconds. When the GA heuristic was used no order missed its deadline. However, when the Min-Slack heuristic was used, 7.1% of the orders missed their deadlines. The GA heuristic also demonstrated a reduction of normalized JIT penalty from 0.058 to 0.00078 compared to Min-Slack. In Figure 12, the left axis is the Missed Order Ratio, which is denoted by *MOR*, while the right axis is the normalized JIT penalty. The value of *MOR* is shown by bars while JIT penalty is shown by lines. Four groups of orders have been compared.

**Situation 2:**  $\times 1.0$  order flow, tighter deadlines.

In the second situation, we decreased the deadlines for all the orders by 200,000 seconds while maintaining other settings the same as in Situation 1. There is no more flexibility to reduce the deadlines. The factory needs to assign any order with at least the amount of processing time estimated under

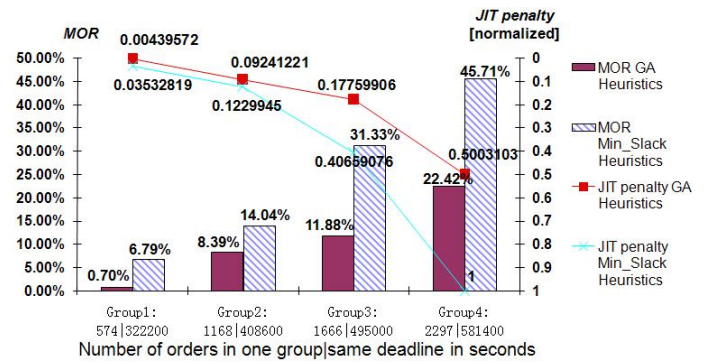


Fig. 14: *MOR* and normalized JIT penalty of order groups under  $\times 1.5$  factory workload.

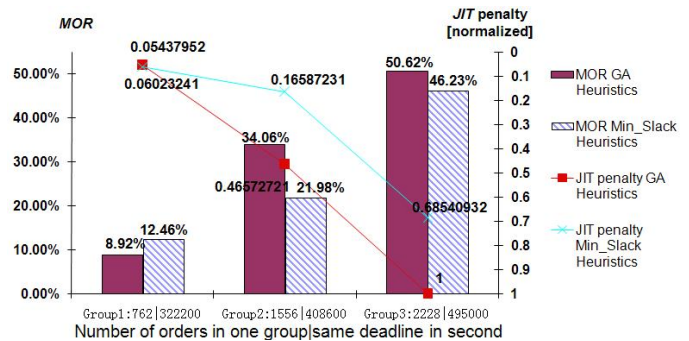


Fig. 15: *MOR* and normalized JIT penalty of order groups under  $\times 2.0$  factory workload.

the best processing condition. This estimated processing time sets the lower bound on an order deadline. The simulation results are presented in Figure 13 as in the same manner of Figure 12. They are discussed in the following subsection.

**Situation 3:**  $\times 1.5$  order flow, normal deadlines.

In the third situation, we only increased the quantity of incoming orders by roughly 1.5 times compared to Situation 1. The other parameters remained the same. The simulation results are presented in Figure 14.

**Situation 4:**  $\times 2.0$  order flow, normal deadlines.

In the last situation, we further increased the quantity of orders by doubling the order quantity in Situation 1. The simulation results are presented in Figure 15. The simulation terminated slightly before 604,800 seconds due to the limited 2G heap space in the software. However, we can still analyze the data obtained from 322,200 to 581,400 seconds.

*C. Discussion of Results*

After analyzing and comparing the results gained from GA and Min-Slack optimization heuristics, the following six conclusions were drawn:

- 1) Under regular situations as in Situation 1, GA has decreased both *MOR* and JIT penalty compared with Min-Slack. The *MOR* has been improved by at least 7% over all the order groups. The value of *MOR* is the primary evaluation parameter of an optimization

technique, because it is directly correlated with factory profits. If the  $MOR$  increases, the extra cost such as upgraded shipping method reduces profits. When the  $MOR$  exceeds a certain threshold, profits can turn into losses.

- 2) Under tighter deadlines (Situation 2) and heavier workloads (Situation 3) than the regular situations, the  $MOR$  and the JIT production worsened for both GA and Min-Slack. However, GA still has lower  $MOR$  and JIT penalty compared with Min-Slack. These situations are not regular situations that occur in a real factory, therefore, they can be viewed as demonstrations of the robustness of GA under unusual severe situations.
- 3) In Situation 4, the workload increased even more, the  $MOR$  exceeded 50%, which indicates that the factory is overloaded. The  $MOR$  and the JIT production deteriorated even more for both GA and Min-Slack. They started approaching each other and the advantage of GA is no longer apparent. The factory reached a saturated status in which improvement by GA was hard to realize. Once the workload increased beyond the factory capacity, other actions should be taken. There are some possible solutions, such as pausing new orders acceptance, extending order deadlines, upgrading and adding resources in the factory, etc.
- 4) Increasing the quantity of resources of one type by adding more resources cannot improve  $MOR$  or JIT production under either heavier-load or tighter-deadline situations. This is because the *throughput* (the quantity of products produced during a certain period) is limited by the *bottleneck processing stages* along the manufacturing lines. A *bottleneck processing stage* refers to a stage whose throughput is much lower than its preceding or following processing stages. For instance, if there is only one binder available in the factory, no matter how fast its preceding processing stages are, the binder's throughput cannot be increased. Tasks will continuously pile up in front of the binder. If the factory only upgrades one bottleneck stage by adding resources to reduce the queue length of a resource, it is more likely that its following stage will become a new bottleneck stage. Therefore, in order to increase the throughput of a factory, the resource capability at every bottleneck stage has to be improved.
- 5) In a real factory, products with the same fulfillment path are about the same size. Resources of the same type are also similar in processing speed. The resource allocation decisions made by GA perform as well as the round-robin strategy. However, when we deliberately modified some resources' capabilities and the sizes of some products, GA showed better improvement in resource balancing than Min-Slack. It is observed that faster machines were utilized more frequently in the case of GA, while in Min-Slack, there was no obvious difference detected since the difference in resources cannot be taken care of by the round-robin strategy.
- 6) In terms of reducing Opportunity Cost ( $OC$ ), GA performed worse than Min-Slack. This is expected since

there is a tradeoff between  $OC$  and  $MOR$ . A reduction in  $MOR$  means that more orders are completed ahead of deadlines. Limited by the factory throughput, not all the orders can be completed right before the deadlines. Some of them have to be scheduled earlier and completed in advance. Therefore,  $MOR$  and  $OC$  cannot be reduced at the same time without improving the factory throughput.

Instead of creating fictitious order flows and factory status, we ran simulations based on the real data from a PSP by introducing reasonable workload and deadline variations as in Situation 2 and Situation 3 to make the results more meaningful, in the sense that they reflect reality more accurately.

## VII. CONCLUSIONS AND FUTURE WORK

We have presented a bottom-up GA-based optimization technique used for real-time and online digital-print factory workflow automation and infrastructure optimization. This technique is integrated into a Ptolemy-based platform that simulates a virtual print factory. The virtual factory runs in a successive order-acceptance and stochastic order-failure environment. It has been shown that simultaneous product-level scheduling and task-level resource allocation for both new and reprocessed orders reduced the number of late orders and improved the quality of JIT production. The robustness and efficiency of the proposed method were compared with baseline methods that are currently utilized in industry.

The proposed bottom-up scheduling scheme also demonstrated greater efficiency in realizing JIT production. The designed fitness function helps the bottom-up GA to approach an optimal scheduling and resource utilization solution simultaneously. Furthermore, by continuously monitoring and updating the resource status in terms of queue length, the proposed optimization techniques were able to perform optimization under the up-to-the-moment factory situation. The dynamically changing resource status also serves as useful diagnostic for human managers. It is crucial for making decisions such as adding or upgrading various types of resources.

Simulation results also shown that, without adding more resources to the factory, the proposed optimization technique reached a limit when the factory continued to accept more orders beyond its capacity. Other methods need to be explored to improve the factory performance even further. We are exploring alternative techniques, such as incremental GA [26], in order to enhance the efficiency of our optimization technique. After more advanced techniques have been integrated, the proposed optimization technique will be implemented on an actual print factory floor.

## ACKNOWLEDGMENT

The authors would like to thank Reischling Press, Inc. (Seattle, WA) for their collaboration, in particular, Susan Jackson, Robert Mitchell and Mark Gustafson.

## REFERENCES

- [1] J. Zeng, S. Jackson, I-Jong Lin, M. Gustafson, E. Gustafson, and R. Mitchell, "Operations Simulation of On-demand Digital Print," To appear in IEEE 13th Joint Int. Conf. Comp. Sci. Info. Tech., Congqing, China, August 2011.

- [2] I-J. Lin, J. Zeng, E. Hoarau, and G. Dispoto, "Next-Generation Commercial Print Infrastructure: Gutenberg-Landa TCP/IP as Cyber-physical System," *Journal of Imaging Science and Technology*, 2010, 54(5).
- [3] J. Spohrer, P.P. Maglio, J. Bailey, and D. Gruhl, "Steps Toward a Science of Service Systems," *IEEE Computer*, 2007, 40(1), pp.71-77.
- [4] S.P. Hoover and G.A. Gibson, "The Future of Print and The Digital Printing Revolution," 31st International Congress on Imaging Sciences, 2010, Beijing.
- [5] H. Kipphan, *Handbook of print media: technologies and production methods*, Springer, 2001.
- [6] S. Mito, T. Ohno *Just-In-Time for Today and Tomorrow*, Cambridge: Productivity Press, Inc., 1986
- [7] M. Agrawal, K. Chakrabarty, J. Zeng, I-Jong Lin and G. Dispoto, "Simultaneous Task Scheduling and Resource Binding for Digital Print Automation," *Proc. IIE. Industrial Engineering Research Conference*, Reno, Nevada, May 21-25, 2011.
- [8] J.M. Buchanna, *Opportunity Cost (2nd Edition)*, The New Palgrave Dictionary of Economics, 2008
- [9] G. J. Stigler, "The Nature and Role of Originality in Scientific Progress," *Economica*, Vol. 22, No. 88, pp. 293-302, Nov. 1995.
- [10] K.R. Baker and G.D. Scudder, "Sequencing with Earliness and Tardiness Penalties: A Review," *Operations Research*, Vol. 38, No. 1, pp. 22-36, 1990
- [11] S. Rai, C.B. Duke, V. Lowe, C. Quan-Trotter and T. Scheermesser, "LDP Lean Document Production - OR Enhanced Productivity Improvements For The Print Industry," *Interfaces*, 39(1), pp. 69-90, 2009.
- [12] J. Zeng, I-Jong Lin, G. Dispoto, E. Hoarau, and G. Beretta, "On-Demand Digital Print Services: A New Commercial Print Paradigm as an IT Service Vertical," 2011 Annual Service Research and Innovation Institute (SRII) Global Conference
- [13] M. Agrawal, Q. Duan, K. Chakrabarty, J. Zeng, I-Jong Lin, G. Dispoto and Y.-S. Lee, "Digital Print Workflow Optimization Under Due-dates, Opportunity Cost and Resource Constraints," *Proc. IEEE International Conference on Industrial Informatics (INDIN)*, 2011
- [14] W. Hopp and M. Spearman, *Factory physics*, 3rd ed. McGraw-Hill/Irwin, 2007.
- [15] M. Holweg, "The genealogy of lean production," *Journal of Operations Management*, volume 25, issue 2, 2007
- [16] R. Anupindi, S. Chopra, S. Deshmukh, J. Van Mieghem, and E. Zemel, *Managing Business Process Flows: Principles of Operations Management (2nd Edition)*, Prentice Hall, 2005.
- [17] M. Nash and S.R. Poling, "Mapping The Total Value Stream: a Comprehensive Guide for Production and Transactional Processes," *Productivity Press*, 2008
- [18] J. Zeng, I-J. Lin, E. Hoarau, and G. Dispoto, "Productivity Analysis of Print Service Providers," *Journal of Imaging Science and Technology*, 2010, 54(6).
- [19] C. Brooks, E.A. Lee, X. Liu, S. Neuendorffer, Y. Zhao and H. Zheng, "Heterogeneous Concurrent Modeling and Design in Java," University of California at Berkeley, 2008.
- [20] K. Chakrabarty, R. Bellamy, G. Dispoto, and J. Zeng, "The Role of EDA in Digital Print Automation and Infrastructure Optimization," To appear in International Conference on Computer-Aided Design (ICCAD) conference, 2011
- [21] G. DeMicheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, New York, NY, 1994.
- [22] K. R. Baker and G. D. Scudder, "Sequencing with Earliness and Tardiness Penalties: A Review," *Ops. Res.* 38, pp. 22-36, 1990.
- [23] M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey," *IEEE Comput.* 27, 17-26, 1994.
- [24] J. C. Bean, "Genetics and Random Keys for Sequencing and Optimization," *ORSA J Comput* 6, pages 154-160, 1994.
- [25] W. M. Spears and K. A. Dejong, "On the virtues of parameterized uniform crossover," *Proc. International Conference on Genetic Algorithms*, 230-236, 1991.
- [26] A.S. Wu, H. Yu, S. Jin, K.-C. Lin and G. Schiavone, "An Incremental Genetic Algorithm Approach To Multiprocessor Scheduling," *IEEE Transactions on Parallel and Distributed Systems*, Vol.15, No.9, pp. 824- 834, Sept. 2004