*Proceedings of the 2014 Winter Simulation Conference*
*A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.*

# JOB RELEASE UNDER DUE DATE CONSTRAINTS IN JOB SHOPS WITH TIME-VARYING PRODUCT MIX

Tao Zhang
Oliver Rose

Universität der Bundeswehr München
Department of Computer Science
Werner-Heisenberg-Weg 39
Neubiberg, 85577, GERMANY

## ABSTRACT

Job shops produce products on the basis of manufacturing orders which specify the due date and the volume. The orders accepted by the shop floor are put into a job pool. The job release decides when to start each job in the pool. It attempts not only to balance this time-varying demand against available capacity, but also manages to meet the due date constraints. The general job release policies, such as output-based or workload-based policies, have poor due date performance. A multi-time-periods release policy is proposed to match the time-varying demand. The due date pressure is distributed to every period. In each time period a near optimal short-term throughput of each product is obtained by an optimization model. The optimization problem is solved by an improved ant colony algorithm. In iteration processes of the algorithm ants are evaluated by the simulation which involves the setup and breakdown of machines.

## 1 INTRODUCTION

Generally, in most job shops products are produced according to manufacturing orders generated by their planning system. For each order a due date is specified and maybe sometimes an earliest start time. An order is connected to a product and often includes many jobs. Usually multiple orders are released to the shop floor at a time. The orders accepted by the shop floor are not directly started but put into a job pool. Dispatchers will decide when each job starts according to the situation of the shop floor. This is the job release problem. Studies on shop floor control are usually focused on the scheduling problem in which the release dates of jobs are assumed to be given and the only problem is the job sequencing at each machine. Thus there is a gap between the planning problem and the scheduling problem. The job release problem bridges the gap. The decisions on the job release can have a significant effect on the shop floor. Therefore, the job release problem should be solved before we carry out the scheduling. For the job release problem the product mix is a key factor and is often time-varying which makes the problem more difficult and complex. The varying mix is attributed to either the different combinations of products or the different throughputs of the same products. The job release attempts not only to balance this time-varying demand against available capacity, but also manages to meet the due date constraints.

According to the information considered in the decision making, the job release can be classified to three types: due date-based job release, workload-based job release, and bottleneck-based release. We will give a short review on them in Section 2. In our case the due dates are assigned to orders rather than the job. It is difficult to calculate the due dates of jobs in the orders according to these due dates. Thus the pure due date-based job release is not suitable for the problem. Even though the other two approaches have been widely used in fixed-demand job shops, difficulties arise in the time-varying-demand job shops. Setting only one constant rate or one target workload level for each product/bottleneck cannot match the

time-varying demand anymore. Moreover, sometimes they assume that the release time of the first job in each order is calculated separately, which leads to poor due date performance. In our study, a multi-time-period release policy is proposed to match the time-varying demand. The calculation of the start time of orders is involved in the policy. The start time of each order and the near optimal throughput of each product in each period can be obtained. The constant rate or target workload level of each product/bottleneck in each period can be calculated according to the throughput.

The paper is structured as follows. In Section 3 we give a detailed description of the problem in the time-varying-demand environment. A general outline of the proposed policy is presented in Section 4. In Section 5 an optimization model is given and an improved ant colony algorithm is used to solve the optimization problem. An application in a job shop environment is reported in Section 6, which also concludes the paper.

## 2 RELATED WORKS

### 2.1 Due Date-Based Job Release

The due date-based job release calculates a release date for each job at decision point according to the due dates and current cycle time estimates, and creates a release time window based on the release date. If the decision point is within the release time window, the related job will be released. The decision is usually made at certain time intervals. There are three common ways to estimate the cycle time. One assumes that the cycle time is proportional to its sum of processing times Conway et al. (2012).They try to find the coefficient of the sum of processing times. One just uses the average cycle time obtained from the historical data of the shop floor. The last believes that the cycle time does not only depend on the sum of the processing times, but also the situation of the shop floor at the decision point. Mahmoodi et al. (1990) used regression analysis method to find the relationship between the total waiting time and the number of jobs on the concerned job's route. The simulation collects the data pair they needed. To determine the due date of a job at a decision point, the obtained regression function is used and the parameter of the function is the number of jobs on the job's route at the decision point. The cycle time equals the total waiting time plus the sum of processing times. Ragatz and Mabert (1988) calculate the cycle time according to the number of operations and the number of jobs on the concerned job's route. Two coefficients are set for these two numbers.

### 2.2 Workload-Based Job Release

The workload is the amount of work that has to be done. It can be measured by the number of in-process jobs or the total processing time of in-process jobs. For the workload-based job release, the decision is made while the workload changed. A releasable job list is included in the method. The releasable job list stores the releasable jobs at the decision point. The jobs in the list are sorted in certain priority sequence. The following are three subtypes of the workload-based job release.

#### 2.2.1 Workload-Based Job Release at the Shop Floor Level

The workload is measured at the shop floor level by terms of the number of in-process jobs or total processing time of in-process jobs in the shop at the decision point. A workload norm is set for the entire shop floor. While a job is finished, the first job in the list will be considered first. If releasing the first job does not cause that the workload exceeds the norm, the job will be released and the workload will be updated. Otherwise, the job will continue to be kept in the pool. Thereafter the second job will be considered. The remaining can be done in the same manner. If the workload exceeds the predetermined workload norm, the rest jobs will not be considered and the release procedure ends. Framinan et al. (2006) developed a dynamic method to determine the norm (card number). The norm is adjusted dynamically according to the throughput of the shop floor at that moment. If the throughput is less than the target

throughput, the norm will increase; otherwise the norm decrease. The lower and upper bounds on the norm restrict the adjustment.

### 2.2.2 Workload-Based Job Release at the Product Level

The workload is measured at the product level by terms of the number of in-process jobs or total processing time of in-process jobs, which belong to the same product, in the shop at the decision point. A workload norm is set for each product. For each product a releasable job list is created as well. While a job belonging to a product is finished, the jobs in the related list will be considered from the first to the last. If releasing a job does not cause that the workloads of the product exceed its norm, the job will be released. With a fixed number of Kanbans (norms) dedicated to each product, Ryan and Vorasayan (2005) use a nonlinear program to evaluate and optimize the allocation of Kanbans to product types. In numerical examples, the allocations identified are similar to those obtained by exhaustive enumeration with simulation, but frequently differ significantly from a naïve allocation according to demand rates. A variant of the model that minimizes the total work-in-process to achieve specified throughput targets yields results similar to a previous heuristic method.

### 2.2.3 Workload-Based Job Release at the Machine Level

The workload is measured at the machine level by terms of the number of jobs or total processing time of jobs, which are/ will be processed on the machine or are waiting before the machine, in the shop at the decision point. A workload norm is set for each machine. While an operation is finished on a machine, the jobs in the list which will visit the influenced machines will be considered in the original sequence in the list. If releasing a job does not cause that the workloads of all related machines exceed their norm, the job will be released. Land and Gaalman (1998) give a general procedure for the workload-based job release. The procedure considers (1) the workload situation on the shop floor in combination with the workload contribution of the jobs, and (2) the relative urgency of the job. The release procedure is built up of two phases, sequencing and selecting. The jobs in the pool are sequenced in order of a planned release date to determine their relative urgency. Urgent jobs have a higher probability to be released, because the jobs are considered in order of planned release dates and the gaps between the workloads and the norms will be largest at the beginning of the release procedure. The selecting phase, choosing jobs that obey the workload norms, is responsible for the load balancing function. The term load balancing refers to maintaining a constant direct load level for each station, which speeds up the throughput in the first place.

## 2.3 Bottleneck-Based Job Release

The bottleneck-based job release is a special case of the workload-based job release at the machine level. It focuses only on the workloads of bottlenecks. The bottleneck principle is converted into a manufacturing control method. Akhavan-Tabatabaei and Salazar (2011) propose a procedure based on trial and error to find the best values for the WIP threshold (norm) in different cases The procedure begins with running the simulation model for no policy case with norm =0 and recording the resulting cycle time and throughput. Then this step is repeated through incrementing the value of norm by one and increasing the value of arrival rate in such a way that the effective arrival rate of cases with policy remain very close to that of the no policy case. The iterations stop when no significant improvement in the cycle time is observed.

# 3 PROBLEM DESCRIPTION

The job release has two tasks. One is to decide the start time of each order, i.e. the start time of the first job or jobs (if several jobs can start at a time). The other task is to determine the start times of the remaining jobs. Many studies assume that the start time of orders equals the earliest start time and then

they have enough slack time to meet the due dates. This usually leads to longer waiting times of finished jobs because they have to wait for other jobs in the order and only all of jobs in the order are completed they can start delivery. The longer waiting time can be problematic. Thus we should start orders as late as possible under the condition that the due dates are met. Figure 1 shows an example where there are five orders in the job pool. Each order includes one product. The latest start time is calculated according to the maximal throughput. If we split the time into several periods according to the product mix, we can see that in some periods only one product is produced and in some periods two or three products are produced. Obviously, if we use the normal approach, we have to specify different policies or different parameters for the policies in the different periods. Someone may believe that just presetting a release policy for each possible situation of the product mix can solve this problem. In that case once the product mix changes, a corresponding policy will be used. However it is not true because the preset policies do not consider the due dates. It is difficult to meet the due dates. The policy or the parameters of the policy are not only dependent on the product mix but also on the due dates.

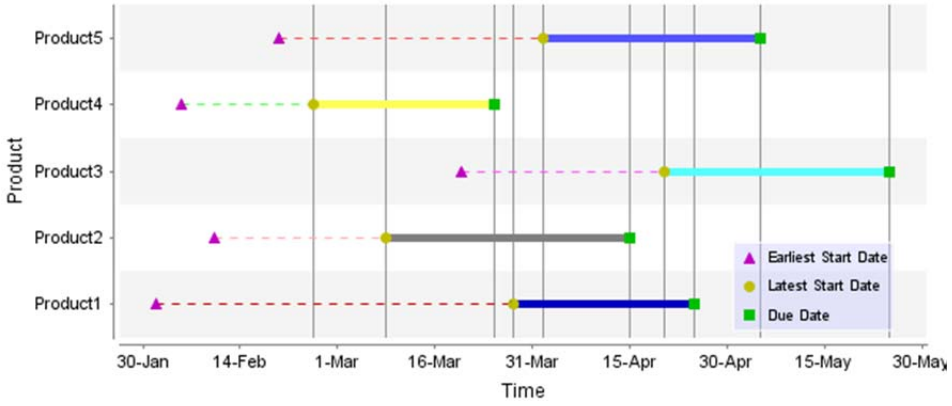


Figure 1: An example of the time-varying product mix.

A reasonable way to solve this problem is to distribute jobs to every period in order to separate due date pressures. In other words, we should specify for each product a short-term throughput in each period. The short-term throughput is dependent not only on the product mix but also on the duration of the period and the time difference from the period to the due dates. When the short-term throughput is determined, we can adopt the normal approach in each period. The parameters of the policy in a period can be determined easily according to the related short-term throughput. So the main problem in the study is to determine the short term throughput in each period. Moreover, because the start time of orders are unknown before, we cannot split the time into several periods at a time. This is very tricky. The start times are obtained from the short-term throughput while the short-term throughput is determined only if the periods are generated according to the start time. Thus, how to split the time into periods is also a very important problem. To sum up, there are two subproblems in our study: finding the time periods which represent different product mixes and determining the short-term throughput in each period according to the product mix and the due dates.

## 4 MULTI-TIME-PERIODS POLICY

As we mentioned before, the time periods cannot be generated at a time. In our approach we find the time periods one by one from the right to the left (backward calculation) beginning from the latest due date of the orders. The start time of the orders are initialized with the latest start time. Once a period is found, an optimization procedure will be started to obtain the short-term throughput for each product included in the period. The start times of related orders are updated according to the short-term throughput. Then we start

to find the next period and perform the same computations. The approach will finish if no more periods can be found. The left part of Figure 2 shows a flow chart of our approach. The remaining part will be introduced in Section 4.

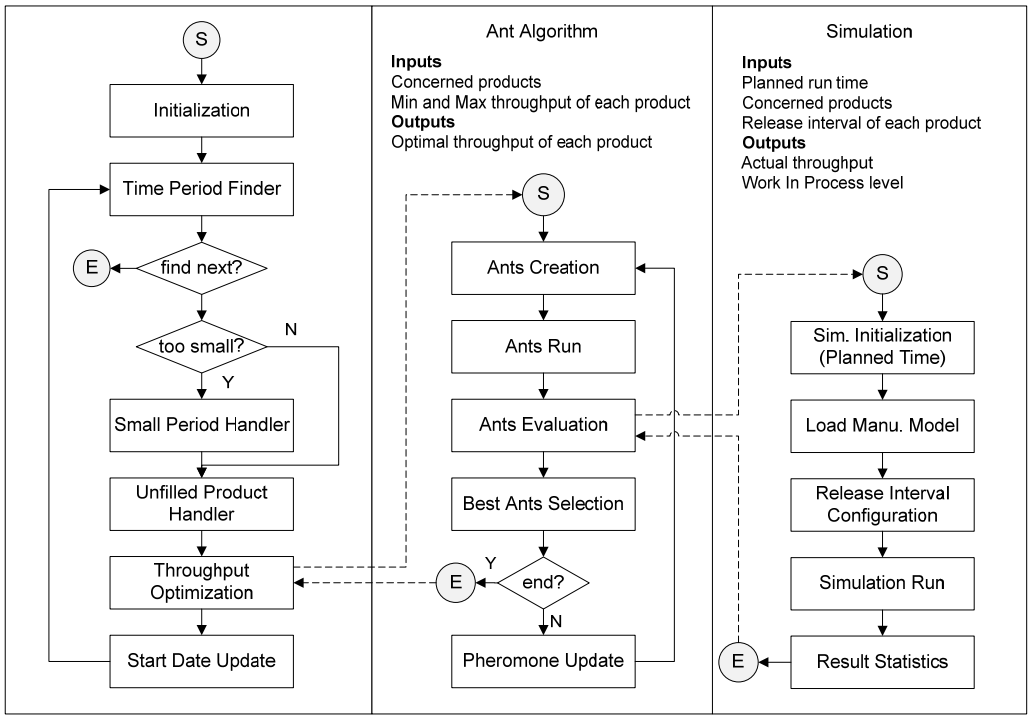


Figure 2: Flow chart of our approach.

### 4.1 Initialization and Time Period Finder

The start time of an order is initialized with the latest start time which is calculated from the maximal throughput. The maximal throughput is the throughput while only the concerned product is produced in the job shop. The latest start time of product $i$ $\tau_i^{LS}$ can be calculated as follows:

$$\tau_i^{LS} = \tau_i^D - N_i / r_i^{Max},$$

where $\tau_i^D$ is the due date of the related order, $N_i$ denotes the number of jobs in the order and $r_i^{Max}$ is the maximal throughput of the product. The maximal throughput is determined by the capacity of the bottleneck machine. It is easy to find out the bottleneck because the bottleneck is fixed when only one product is produced.

A time period finder is responsible to generate the time period. At the beginning the end time of the first period is set to the latest due date of orders. The start time or the due date which is closest to the end time of the periods will be the start time of the period. When we try to find the next time period, the start time of the period will become the end time of the next period. The rest will be carried out in the same manner. The start time of the *k*-th period $\psi_k^S$ is computed by

$$\psi_k^S = \min(\min(t_i^S), \min(t_i^D))$$
$$t_i^S = \psi_k^E - \tau_i^S, \tau_i^S < \psi_k^E, \psi_k^E = \psi_{k-1}^S, i \in P$$
$$t_i^D = \psi_k^E - \tau_i^D, \tau_i^D < \psi_k^E, \psi_k^E = \psi_{k-1}^E, i \in P,$$

where $\psi_k^E$ is the end time of the *k*-th period, $\tau_i^S$ and $\tau_i^D$ are the start time and the due date of product *i*. *P* denotes the set of all products. $t_i^S$ and $t_i^D$ are the time differences from the end time of the period to the start time and the end time of product *i*.

### 4.2 Small Period Handler and Unfilled Product Handler

Following the above steps, we may find very small period durations. To determine the short-term throughput in such small periods makes no sense. We consider a period too small when the duration is less than ten times the longest release interval of products. Once a small period is found, we will give up the found start time immediately and looking for next earlier starting time. If the new duration is still small, just continue until the duration is longer than the limitation (as shown in Figure 3 (a)).

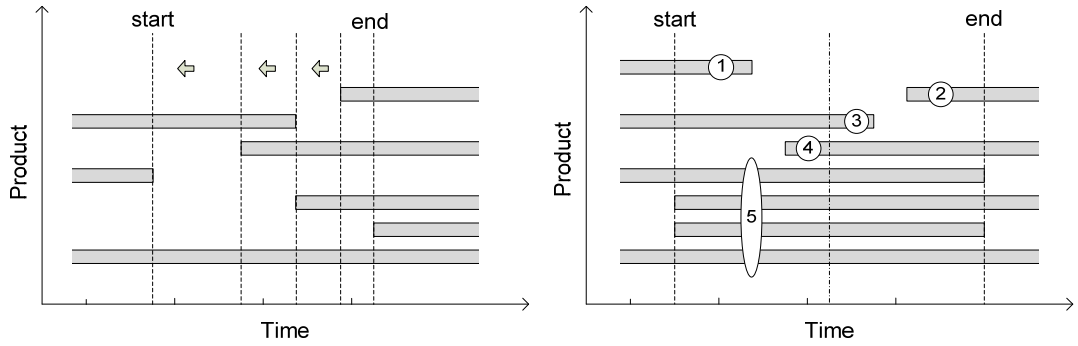


Figure 3: Small period handler (a) and unfilled product handler (b).

The small period handler leads to a new problem, shown in Figure 3 (b). There are products which do not fully fill up the period (marked as 1, 2, 3, and 4). The fully filled products (marked as 5) will be chosen and taken to the optimization phase. The problem is how to deal with the unfilled products. We select the near fully filled products (1, 2) to the optimization phase too. The less filled products (3, 4) will not be chosen. In this period the product 2 will adopt its throughput in the previous period and the product 1 will use the throughput from the next period.

### 4.3 Throughput Optimization and Start Time Update

Once a time period is found, the short-term throughput will be obtained through an optimization procedure which will be introduced in Section 5. The optimization is carried out for the chosen products in the period according to the range (min and max values) of the throughputs of each chosen product. The lower bound of the short-term throughput of product *i* in the *k-th* time period is calculated as follows.

$$r_{i,k}^L = (N_i - \sum_{j=1}^{k-1} r_{i,j}(\psi_j^E - \psi_j^S) - r_i^{Max}(\psi_k^S - \tau_i^{ES}))/(\psi_k^E - \psi_k^S),$$

where $r_{i,j}$ is the short-term throughput of product *i* in the *j*-th period. $\psi_j^S$ and $\psi_j^E$ are the start time and the end time of the *j*-th period. $\tau_i^{ES}$ is the earliest start time of product *i*. The lower bound ensures that the remaining jobs of the product can be finished in the remaining time with the maximal throughput. The upper bound of the range cannot be the maximal throughput. If so there will be not enough capacity for other products to even reach their lower bound throughput. Thus the upper bound should be the throughput while other products are produced at the lower bound pace and the remaining capacity is fully utilized by the product. Even though the bottleneck of the product may change, the upper bound is still easy to determine because the throughput of the remaining products is fixed to the lower bound. Only if

there is just one product in the period, the upper bound can be the maximal throughput. In this case the lower bound is already the maximal throughput. Thus the maximal throughput is the optimal throughput in this case.

The start time of each chosen product will be updated after we know the short-term throughput in the period. The short-term throughput is only used for this period. In the later time periods the maximal throughput is still effective. The start time of the product *i* is updated as follows.

$$\tau_i^S = \tau_i^{S'} + (r_i^{Max} - r_{i,k})(\psi_k^E - \psi_k^S) / r_i^{Max},$$

where $\tau_i^{S'}$ is the old value. The other variables have been explained before. For the unfilled product types 3 and 4 (in Figure 3), we just need to change the duration to the real one. For the unfilled product type 2 (in Figure 3), we use the throughput from the previous time period.

$$\tau_i^S = \tau_i^{S'} + (r_i^{Max} - r_{i,k-1})(\psi_k^E - \tau_i^{S'}) / r_i^{Max}$$

For the unfilled product type 1 (in Figure 3), we do not update its start time in this period but in the next period because the short-term throughput is the throughput in the next period, .

$$\tau_i^S = \tau_i^{S'} + (r_i^{Max} - r_{i,k+1})(\tau_i^D - \psi_k^S) / r_i^{Max}.$$

## 5 OPTIMIZATION FOR EACH TIME PERIOD

Once a time period is found, we try to determine a reasonable short-term throughput for each chosen product under the constraints of the lower and upper bounds that we calculated before. This work is carried out by means of an optimization procedure. In this section we will introduce the optimization procedure in detail.

### 5.1 Optimization Model

The first step is to create an optimization model. The optimization model is usually composed of an objective, several decision variables and some constraints. Naturally, the short-term throughputs of products are the decision variables. The lower and upper bounds of the throughputs are one kind of constraints. The objective in our study has two parts: maximizing the overall throughput and minimizing the average WIP level (see the following equations). They are integrated into one maximizing objective by means of two coefficients.

$$\max \quad f(r_{i,k}^{sim}, \kappa_{i,k}^{sim}) = \alpha \sum_{i=1}^{m} w_{i,k} r_{i,k}^{sim} + \beta / \sum_{i=1}^{m} w_{i,k} \kappa_{i,k}^{sim},$$

$$\text{where } w_{i,k} = r_i^{Max} / (\tau_i^{S'} - \tau_i^{ES}) \text{ and } r_{i,k}^{sim}, \kappa_{i,k}^{sim} \xleftarrow{sim} v_{i,k},\ v_{i,k} = 1 / r_{i,k}.$$

The weight $w_{i,k}$ represents the urgency of the product *i* in the *k*-th period. *m* denotes the number of the chosen products in the period. The short-term throughput $r_{i,k}$ is not considered in the objective function directly. It is converted to the release time interval $v_{i,k}$ and then input into the manufacturing simulation. $r_{i,k}^{sim}$ is the modified short-term throughput. $\kappa_i^{sim}$ is the average WIP level of product *i*. Both of them are obtained from the simulation. $\alpha$ and $\beta$ are two coefficients which represent the importance of the throughput and the WIP level in the objective function. Here, we assume $a = \beta = 0.5$.

### 5.2 Ant Colony Algorithm

The ant colony algorithm is used to solve the optimization model. The algorithms are stochastic search procedures. Their central component is the pheromone model, which is used to probabilistically sample the search space (Dorigo and Blum 2005). Because the optimization model is a continuous model, but the

ant colony algorithm is only suitable for a discrete model, the continuous decision variables have to be converted to discrete variables first (Liao et al. 2014). We normalize all variables to [0,1) and specify for each variables a number of digits after the decimal point. Each decimal place can be one digit from 0 to 9. Thus a network can be created as shown in Figure 4. The start node and the end node have no real meaning. Each of other columns represents a decimal place of one variable. Figure 4 shows an example having two variables and three decimal digits for each variable. Thus the continuous problem turns into a discrete routing problem in the network.

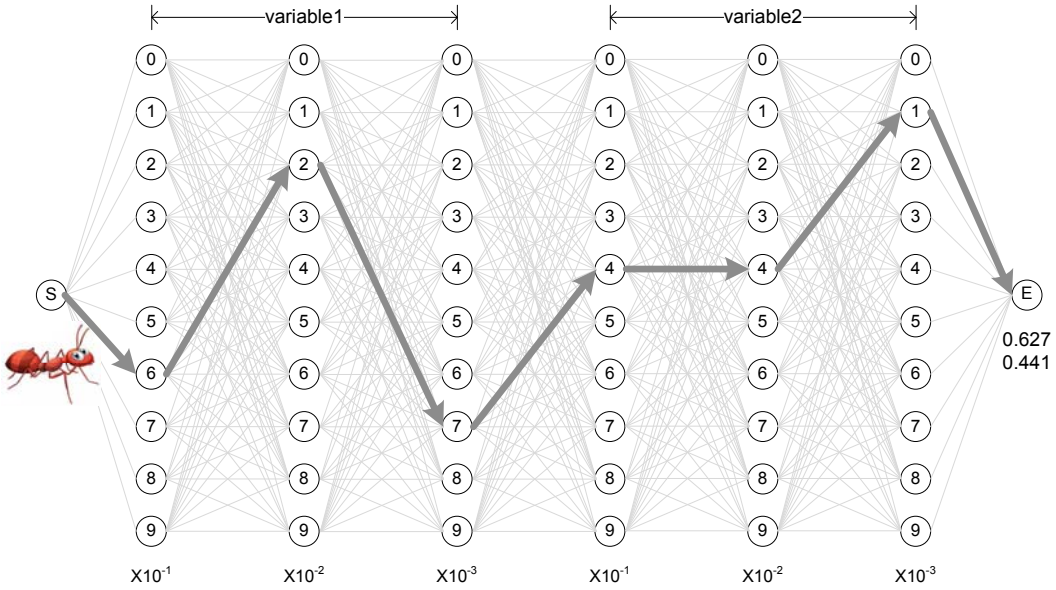


Figure 4: Discretization of the continuous problem for the ant colony algorithm.

Ants will run on the network and leave the pheromone on the sides. The pheromone also can be evaporated as time goes on. Which side the ant will take depends on the volume of the pheromone on the sides. The probability to select one side is directly proportional to the volume of the pheromone (roulette wheel selection). The path that an ant goes through denotes a solution. We update the pheromones only after the ant finishes the run. The increased pheromone level is related to the evaluation results of the ant. The evaporation rate of pheromones is fixed. When more and more ants go through the network, there will be one path with very high pheromone levels and most of ants run on the path. This path will be the optimal solution. In our approach, the short-term throughputs are normalized according to their lower and upper bound. In order to speed up the convergence we form batches of ants. For each batch we select several best ants and update the pheromones according to these ants' performances. The procedure is shown in the center part of Figure 3. The pheromone update is performed as follow.

$$\lambda = (1-\rho)\lambda_0 + f(r_{i,k}^{sim}, \kappa_{i,k}^{sim})Q,$$

where $\lambda$ and $\lambda_0$ are the new and old value of the pheromone. $\rho$ is the evaporation rate. $Q$ denotes the importance of the objective.

### 5.3 Manufacturing Simulation

The manufacturing simulator we proposed in another paper (Zhang and Rose 2012) is used to evaluate each ant. The short-term throughputs that each ant implies are converted to release time intervals. During the simulation, jobs are released according to the release time intervals. Setups, breakdowns and maintenances are also involved in the simulation. The dispatching rule FIFO is adopted by all machines. The simulation outputs the throughputs of the products and the average WIP levels which will be used in the objective function to calculate the ant's performance. When the optimization finishes, we replace the optimal short-term throughputs with the throughputs from the simulation which will be adopted by the corresponding time period and used to update the start times of the products. Because the simulation model is stochastic, we try to evaluate as many ant simulation runs as possible to obtain significant results. But due to the run time limitation of the approach, the simulation run times are limited. The minimal run times can be determined by the average deviation of the simulation outputs. The average deviation is decreasing while the run times increase. A norm is preset in advance. Once the deviation reduces to the norm, we believe that the mean values of the simulation outputs are almost true value.

## 6 EXPERIMENTS AND CONCLUSION

### 6.1 Experiments

We carried out experiments for a job shop. The job shop has 12 machines with different functions. The machines break down randomly, and the recovering time is dependent on the type of the breakdowns. At the beginning of February there are five orders accepted in the job pool. Each order is related to a product. Different products have different processing flows and different cycle times. We put these fiver orders into our consideration.

In the ant colony algorithm, the number of decimal places is 4; $\rho$ is 0.23 and Q is 1. The result is shown in Figure 5. There are 8 time periods found. The density of the vertical lines in the periods represents the value of the short-term throughput. In the denser periods the short-term throughput is higher. Besides, from the diagram we can see that only one product is started at the earliest start time. The other products are all started later. The time that we saved will result in an inventory cost reduction.

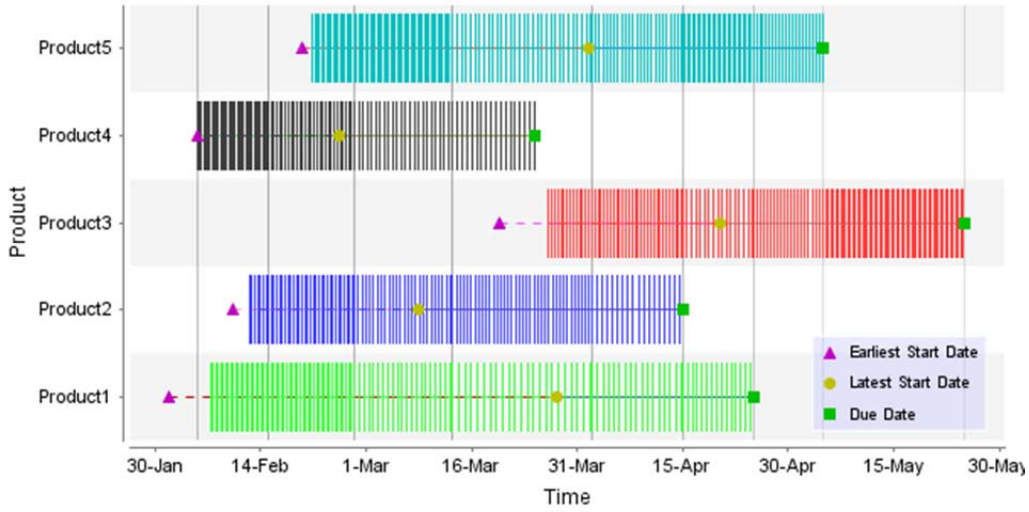


Figure 5: Results of the experiment: time periods and the short-term throughput of each product.

A simple way to use the short-term throughputs we obtained is to calculate the release interval or the target WIP level for each product at each time period. Then we can use the constant interval policy (CONINT) or the constant WIP level policy (CONWIP) to release the jobs. It is nature that the average WIP level and the average cycle time of each product is varying among different time periods. The bottlenecks are also varying among different time periods due to the varying product mix. Except CONINT and CONWIP, how to more effectively use the short-term throughputs to release the job, will be our next work.

### 6.2 Conclusion

The proposed approach divided the manufacturing time interval into several time periods according to the different product mixes. For each period a short-term throughput is specified to each product. The dividing process is carried out backwards. Therefore, it is straightforward to meet the due dates. The start times that we finally obtained were usually later than the earliest times. The time that we saved will result in an inventory cost reduction. The short-term throughputs are from the results of the stochastic simulation. They are robust enough to adapt to the unexpected events. In the optimization model, the urgency of each product is involved in the objective, which can guarantee that the products start later than the earliest start time. The best ants are selected to update the pheromone to speed up the convergence. The shortcoming of our approach is the high time consumption because the simulation had to run several thousand times.

## REFERENCES

Akhavan-Tabatabaei, R., and C. F. R. Salazar. 2011. "Effective WIP Dependent Lot Release Policies: A Discrete Event Simulation Approach." In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, 1976-1985. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Conway, R. W., W. L. Maxwell, and L. W. Miller. 2012. *Theory of Scheduling.* Dover Publications.

Dorigo, M., and C. Blum. 2005. "Ant Colony Optimization Theory: A Survey." *Theoretical Computer Science*. 344 (2): 243-278.

Framinan, J. M., P. L. González, and R. Ruiz-Usano. 2006. "Dynamic Card Controlling in a Conwip System." *International Journal of Production Economics* 99 (1): 102-116.

Land, M. J., and G. J. Gaalman. 1998. "The Performance of Workload Control Concepts in Job Shops: Improving the Release Method." *International Journal of Production Economics*. 56: 347-364.

Liao, T., T. Stützle, M. A. Montes de Oca, and M. Dorigo. 2014. "A Unified Ant Colony Optimization Algorithm for Continuous Optimization." *European Journal of Operational Research*. 234 (3): 597-609.

Mahmoodi, F., K. J. Dooley, and P. J. Starr. 1990. "An Evaluation of Order Releasing and Due Date Assignment Heuristics in a Cellular Manufacturing System." *Journal of Operations Management* 9 (4): 548-573.

Ragatz, G. L., and V. A. Mabert. 1988. "An Evaluation of Order Release Mechanisms in a Job-Shop Environment." *Decision Sciences* 19 (1): 167-189.

Ryan, S. M., and J. Vorasayan. 2005. "Allocating Work in Process in a Multiple-Product Conwip System with Lost Sales." *International Journal of Production Research* 43 (2): 223-246.

Zhang, T., and O. Rose. 2012. "Developing an Agent-Oriented Parallel Simulator for Production Processes." In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, Poster 135. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

## AUTHOR BIOGRAPHIES

**TAO ZHANG** is a Ph.D. student working on production planning and scheduling at the Department of Computer Science of the Universität der Bundeswehr München, Germany. From 2007 to 2009 he received his Master in metallurgical engineering with the subject of production planning and scheduling in iron and steel industry from Chongqing University, China. He is involved in modeling and simulation of complex system and intelligent optimization algorithms. His email address is tao.zhang@unibw.de.

**OLIVER ROSE** holds the Chair for Modeling and Simulation at the Department of Computer Science of the Universität der Bundeswehr, Germany. He received a M.S. degree in applied mathematics and a Ph.D. degree in computer science from Würzburg University, Germany. His research focuses on the operational modeling, analysis and material flow control of complex manufacturing facilities, in particular, semiconductor factories. He is a member of IEEE, INFORMS Simulation Society, ASIM, and GI, and has been the General Chair of WSC 2012. His email address is oliver.rose@unibw.de.