A Scheduling Heuristic for Job Shop Manufacturing System

1Chih-Hung Tsai, 1Yi-Chan Chung and 2Chi-Ming Chen

1 Department of Industrial Engineering and Management
Ta-Hwa Institute of Technology
1 Ta-Hwa Road, Chung-Lin
Hsin-Chu, Taiwan, ROC
Tel: +886-3-5430-466
Email: ietch@thit.edu.tw

2 Mechanical Industry Research Laboratories, ITRI
Bldg. 21, 195 Sec. 4, Chung-Hsing Rd.,
Chutung, Hsin-Chu, Taiwan 310, ROC

Abstract

In the past, many production-scheduling methods were developed to achieve high due-date and short manufacturing lead times performance. Among them, DRUM-BUFFER-ROPE (DBR) has been recognized as one of the excellent solutions. However, there are some implementation constraints (e.g., time buffer determination, setup ratio evaluation, non-CCR subordination etc.) limit its implementation performance. To overcome these limits and enhance the DBR implementation, we present a novel scheduling heuristic algorithm for job shop manufacturing system based on Capacity Constraint Resource (CCR). The proposed algorithm consists of four stages: (1) schedule development for CCR (DRUM), (2) CCR setup time savings to decrease mean tardiness, (3) CCR setup time savings to maximize CCR throughput, and (4) development of a hybrid scheduling method to minimize total flowtime and the work-in-process (WIP) inventory. Also, an illustrative example demonstrates the feasibility of the proposed method.

Keywords: Capacity Constraint Resource (CCR), DBR, Hybrid Scheduling, TOC.

1. Introduction

DRUM-BUFFER-ROPE (DBR) (Goldratt and Fox 1986, Goldratt 1990a, Fogarty et al. 1991, Schragenheim 1991), a newly emerging scheduling concept based on theory of constraints (TOC) (Goldratt 1990b, Schragenheim and Ronen 1990), views production scheduling and the use of capacity in such a perspective that has significantly altered the manner in which many production and inventory professionals think about how to utilize resources properly, particularly in reducing setup times. The theory of constraints is a new philosophy that has a lot in common with Just-in-Time (JIT). The constraint is defined as anything that limits a system from achieving higher performance relative to its goal. The TOC approach is centered on a five-step process, which Goldratt refers to as the five focusing steps. These five focusing steps of TOC are: (1) identifying the system constraint, (2) deciding how to exploit the system constraint, (3) subordinating everything else to the above decision, (4) elevating the system
constraint, (5) identifying the new system constraint if the constraint is broken in step 4; inertia should not be allowed to become the system constraint. In exploiting the constraint, the DBR scheduling technique and BUFFER management are used. In finding ways to elevate the constraint, the techniques of effect-cause-effect and the cloud diagram often are useful.

In DBR, resources belong to categories of constraint and non-constraint. A constrained resource's productive capacity represents our most valuable asset; a non-constrained resource's capacity represents somewhat more capacity than deemed necessary to fulfill market demand. If such a case arises, any process improvement designed to improve the system throughput should have a constraint as its target. Otherwise, our time is wasted, not saved. Therefore, setup time savings should be considered differently in a constrained resource than in non-constrained ones. Setup time savings in the constrained resource imply that more system throughput will be gained, thereby accounting for the emergence of a great need for setup time savings in the constrained resource. However, setup time savings for non-constrained resources will merely contribute to idle time and will not add to the system throughput; thus, an hour saved at a non-constraint is just a mirage. Therefore, non-constrained machines should focus on ways to subordinate the constrained machine, minimize the total flowtime, and reduce total WIP inventory.

In DBR, the DRUM is defined as a schedule for the constrained machine. It is used to maximize the constrained machine's available time. The BUFFER is defined as a protection of the constrained machine against adjacent disruptions. Disruptions might stem from a variety of reasons: machine breakdowns, absenteeism, fluctuations in setup times, unreliable vendors, scrap, or merely the unavailability of a certain resource because it is used in other jobs at that particular time. The BUFFER is the time provided for the parts to reach the protected area. The protected area includes the DRUM, the due-dates and assemblies of constraint parts with non-constraint parts. This is a time mechanism that not only offsets those things going wrong, but also determines the lead time for products from releasing. The BUFFER is equal to the processing time plus the setup time plus an estimate of the aggregated amount of protective time deemed necessary to ensure that the product arrives to the BUFFER origin when required. Three areas (the BUFFER origins) need protection (Figure 1): (1) a Shipping BUFFER to ensure that parts are delivered to the customer on time, (2) a Constraint BUFFER to ensure maximum utilization of constraining machine time, and (3) an Assembly BUFFER to ensure that parts processed on the constraint do not wait in the assembly operation before parts from other non-constrained machines arrive. The ROPE is defined as a schedule for releasing raw materials to the shop floor. It is the synchronization mechanism for the other machines and consists of the release schedule for raw materials. Technically, the ROPE equals the constraint schedule date minus the BUFFER time. Also, the release of raw materials determines the timing for parts being processed on the non-constrained machines.
Setup time savings for the constrained machine in DBR is achieved by performing a setup ratio algorithm. Setup ratio denotes the extent into the future that it may search for similar batches to move them to save setups. Those batches are then moved earlier and placed immediately after the similar batches even though another batch is made late or later. To use this algorithm, a user must specify a ratio of how many hours and minutes ahead to look for a batch to move in order to save an hour of setup. However, DBR does not address how to determine the setup ratio (Schragenheim 1991). Subordination of non-constrained machines is achieved by releasing the parts at the time of the constrained machine’s starting time minus the BUFFER time (CCR or assembly BUFFER). This subordination concept adequately protects the constrained machine. However, the total flowtime and WIP are negatively affected (Schragenheim and Ronen 1990).

Both of these shortcomings of DBR limit its application in a job shop manufacturing system to overcome the limitations of DBR in a job shop manufacturing environments. Figure 2 illustrates the four-stage scheduling heuristic algorithm. As indicated in this figure, stage 1 is schedule development for CCR. This stage adopts DBR DRUM schedule algorithm to create DRUM schedule for a constrained machine. However, this study uses the job's total processing time after CCR substitute for shipping BUFFER to create the RUIN (infinite capacity scheduling for CCR). Stage 2 is CCR setup time savings for decreasing mean tardiness of a constrained machine (or DRUM schedule). Stage 3 is CCR setup time savings for maximizing throughput of a constrained machine. Stage 4 is hybrid scheduling method development (Figure 3) for non-constrained machines to minimize total flowtime and WIP inventory. A Minimum-In-Processing-Time (MIPT) concept is created herein to minimize total mean flowtime and WIP inventory. The scheduling heuristic algorithm is intended to balance due-date performance and setup time savings, achieve maximum system throughput, and minimize the total flowtime and WIP level.
2. The Scheduling Heuristic Algorithm

As Figure 2 illustrates, the scheduling heuristic algorithm consists of four stages. Details of each stage are described as follows:

Stage 1: DRUM Creation

DRUM development herein resembles DRUM development of DBR. The difference is that the job's total processing time after CCR replaces the shipping BUFFER defined in DBR. In performing this function, the stage follows these steps:

Step 1: Develop RUIN (infinite capacity scheduling for CCR)

The RUIN schedule is derived directly from the sales order due-date \( d_i \) minus the job's total processing time after CCR \( B_i \). The nickname "RUIN" is given because of the visual association with blocks of an arbitrary length put on top of each other. However, the RUIN represents the ideal schedule because of the perfect resolution of the conflicts between the due-date and CCR's schedule. No concession for CCR's capacity has been made. Conflicts of CCR's own capacity are clearly observed, as shown in
Figure 4a (assuming that we have only two units of M machines, M(1) and M(2)). Therefore, batch nos. 4, 9, 10, and 14 are an unrealistic schedule, and RUIN must be rescheduled.

**Step 2: Develop DRUM** (finite capacity scheduling for CCR)

How should this conflict be resolved so that it simultaneously but not negatively affects the due-dates? What is left is to move some CCR's operation backwards in time to resolve the capacity conflicts. Since in this study any batch is placed no later than its original timing on the RUIN, the latest order should first be handled and then systematically move backwards in time (the larger $A_i$ is selected to be the latter one). The first batch to be placed is batch no 4 (Figure 4b). The second batch to be placed is batch no 9, as placed before batch no 7 (Figure 4c). Continuing the same process yields the results in Figures 4d and 4e. However, all of the conflicts have not yet been resolved. The capacity conflict is still shown in a different form. Batch no 14 is scheduled to start before time zero. Therefore, DRUM must be modified.

**Step 3: Modify DRUM**

Many options are not available. The only alternative is to push the batches, which violated the realistic requirement of not being scheduled in the past, to the present. Some other batches may be pushed into the future, thereby becoming "late batches." The setup time savings method can be used to improve system performance. Although setup time savings for CCR can provide more CCR time for further use, except for CCR time savings, setup time savings can either decrease mean tardiness or maximize CCR throughput. For the former, batches are moved backwards to merge with the same part only if the movement does not make the intermediate batches late or later. For the latter, however, batches are moved backwards to merge with the same part only if the total tardiness save of those batches moving backwards is greater than or equal to the total tardiness increase of those batches moving forwards. Stages 2 and 3 include detailed algorithms for both setup time savings.

![Figure 4a: CCR's RUIN](image-url)
Stage 2: Setup Time Savings for Decreasing Mean Tardiness

This stage moves batches around to save setups, by placing batches of the same part/operation consecutively on the same unit of the constraint. No batches will be moved if their movement does not decrease the overall tardiness. In performing this function, the stage follows these steps:

Step 1: Select the part from DRUM and set it as the original batch.
Step 2: Lookahead at the same part from the original batch. If the part is obtained, then set it as the locked batch and goto step 3; otherwise, goto step 6.
Step 3: If either there is at least one late batch later than the current location of the locked batch or the locked batch is late, then goto step 4; otherwise, goto step 6.
Step 4: If no batch, currently between the original batch and the locked batch, will become late or later because of the batch merged, then goto step 5; otherwise, goto step 6.

Step 5: Calculate the batch’s starting and finishing time. Lookahead at the same part from the locked batch. If the part is obtained, then set it as the locked batch and goto step 3; otherwise, goto step 6.

Step 6: Lookahead at the adjacent different part from the original batch. If the part is obtained, then set it as the original batch and goto step 2; otherwise, end.

Stage 3: Setup Time Savings for Maximizing Throughput

This stage moves batches around to save setups with regard to how it affects the total tardiness of batches. No batches will be moved if their movement does not increase the overall throughput. In performing this function, the stage follows these steps:

Step 1: Select the part/operation from DRUM and set it as the original batch.

Step 2: Lookahead at the same part from the original batch. If the part is obtained, then set it as the locked batch and goto step 3; otherwise, goto step 6.

Step 3: If total tardiness saved is greater than or equal to the total tardiness increased because of the batch merged, then set the locked batch as the benchmark batch and goto step 4; otherwise, goto step 5.

Step 4: Lookahead at the same part from the locked batch. If the part is obtained, then set it as the locked batch and goto step 3; otherwise, goto step 5.

Step 5: If the benchmark batch is null, then goto step 6; otherwise, merge batch (same part as the original batch) between the original batch and the benchmark batch, calculate the batch’s starting and finishing time.

Step 6: Lookahead at the adjacent different part from the original batch. If the part is obtained, then set it as the original batch and goto step 2; otherwise, end.

The optimal DRUM can subsequently be formed.

Stage 4: Hybrid Scheduling

Adopting a hybrid scheduling method (Hastings and Yeh 1990), that enables individual jobs to be scheduled either forward or backward in one schedule can develop a detailed non-CCR schedule. The hybrid scheduling method for a given job generally involves three phases: (1) generating an initial schedule by backward or forward scheduling, (2) determining the core operation based on the initial schedule, and (3) adjusting the job’s other operations by rescheduling (forwards or backwards) them close to the core operation. In backward scheduling, subsequent operations of the core operation are scheduled forwards from its scheduled finish time. In forward scheduling, prior operations of the core operation are scheduled backwards from its scheduled start time.

Figure 5 illustrates how the minimization of total flowtime concept can be achieved by the hybrid scheduling method. Figure 5(a) considers the forward schedule of job H. Job H has three operations, i.e., H1, H2, and H3, which must be sequentially performed at workcentres M1, M2, and M3. An unnecessary gap occurs between operations H1 and H2. Similarly, an unnecessary gap occurs between operations J2 and J3 in Figure 5(c), which is an example of backward scheduling for job J. Figure 5(b) presents a preferable schedule for job H in Figure 5(a) in terms of minimizing the in-process time. Also, the operations of job H are scheduled close to operation H2 rather than operation H1 in Figure 5(a). As for job J in Figure 5(c), a better schedule would be the one shown in Figure 5(d), where the
operations of job J are scheduled close to operation J2 rather than operation J3 in Figure 5(c).

Figure 5: (a) Forward Scheduling for Job H; (b) Hybrid Scheduling for Job H

Figure 5: (c) Backward Scheduling for Job J; (d) Hybrid Scheduling for Job J

In this study, we employ the optimal DRUM from Stage 3 as the set of core operations, in which above three hybrid scheduling phases are combined into one major phase: adjusting the prior/subsequent operations of the job's CCR by scheduling them backwards/forwards close to the core operation (DRUM). Consequently, the computational time involved in determining the core operation can be reduced. The main procedure is described as follows (Note: Figure 6 depicts the detailed algorithm of hybrid scheduling):

**Step 1:** Select CCR priority job $i$ from DRUM until the end of DRUM. Then the operations before CCR (core operation) are scheduled finitely backwards from the CCR start time. Considering factors such as overtime, subcontract, and offload can solve any batches scheduled to start before time zero.

**Step 2:** Select CCR priority job $i$ from DRUM until the end of DRUM. Then the operations after CCR (core operation) are scheduled finitely forwards from the CCR finish time.

**Step 3:** Schedule the free orders backwards from its due-date. The priority rule is EDD (Earliest Due-Date). The free order means that order's operations will not go through the bottleneck machine.

**Step 4:** Calculate the job's minimum-in-processing time (i.e., the job's finishing time minus the job's releasing time).

3. An Illustrative Example

The following example demonstrates the effectiveness of the scheduling heuristic algorithm in more detail. It is derived from DISASTER (Schragenheim 1991). The job shop manufacturing system uses six different machines (M1, M2, M3, M4, M5, and M6). Table 1 summarizes product's demand data (product B, D, F, and H) and Table 2 summarizes the workcenter data. Figure 7...
presents the routing layout and Figure 8 presents the detailed processing data. In Figure 7, operation flow is bottom up. Each box is an operation, done by a resource denoted by its capital letter. Numbers denote average processing time per part. Each operation is recognizable by its column/row, i.e., the leftmost M3 operation, being in column A and row 5, is A5. Product B is made from raw materials A and C. Machine M1 processes raw material A for 5 minutes on average, and delivers the part to the M2 machine for 18 minutes, etc. Resource M5 is an assembly machine, each assembly operation being a simple 1:1 assembly. There are 10 pieces of WIP past A3 operation (ready to be processed by M3 at A5). Another 10 lie past the C5 operation, and another 16 lie past the E3 operation, etc.

**Stage 1: DRUM Creation**

Figure 8 also reveals that M1, M2, M3, M4, M5, and M6 machine loading are 1028, 1887, 1789, 2736, 1796 and 1874 minutes. M4 machine loading (2,736 minutes) is greater than others are. Therefore, we select it as the bottleneck machine.

**Step 1: Develop RUIN (infinite capacity scheduling for CCR)**

The RUIN schedule is derived directly from the sales order due-date ($d_i$) minus the job's total processing time after CCR ($B_i$). For instance, part E5(H-12-2)'s finishing time is equal to 468 (960 - 492) minutes, Figure 9a summarizes those results.

**Step 2: Develop DRUM (finite capacity scheduling for CCR)**

Herein, some of M4's operations are moved backwards in time to resolve the capacity conflicts without jeopardizing the due-dates. For instance, part G5(F-10-3) is moved backwards and placed before part E5(F-10-7). Figure 9b summarizes those results.

**Step 3: Modify DRUM**

As indicated in Figure 9b, part E5(H-12-2) is scheduled to start before time zero (386 minutes before time zero). Therefore, all parts must be pushed forwards. Figure 9c summarizes those results.

**Stage 2: Setup Time Savings for Decreasing Mean Tardiness**

First, we select part E5(H-12-2) from DRUM, set it as the original batch, and then lookahead at the same part. Consequently, no batch is qualified. Second, we select part G5(H-12-2) from DRUM, set it as the original batch, and then lookahead at the same part G5(F-10-3) from the original batch. Part G5(F-10-3) is not late, but parts B-16-4, H-8-5, H-8-6, and F-10-7 are late. After parts G5(H-12-2) and G5(F-10-3) have merged, no batch becomes late or later. Also, the final scheduling time of DRUM is 3,456 minutes, thereby saving 60 minutes. Figure 9d summarizes the merged results.

**Stage 3: Setup Time Savings for Maximizing Throughput**

First, we select part E5(H-12-2) from DRUM, set it as the original batch, and then lookahead at the same part E5(F-10-3) from the original batch. Part E5(F-10-3) is late. After parts E5(H-12-2) and E5(F-10-3) have merged, total tardiness saved is greater than or equal to the total tardiness increased. Concurrently, parts C5(B-9-3) and C5(B-16-4) have also merged. Also, the final scheduling time of DRUM is 3,336 minutes, thereby saving 120 minutes. Figure 9e summarizes the merged results.

Second, we select part E5(H-8-5) from DRUM, set it as the original batch, and then lookahead at the same part E5(H-8-6) from
the original batch. Part E5(H-8-6) is late. After parts E5(H-8-5) and E5(H-8-6) have merged, total tardiness saved is greater than or equal to the total tardiness increased. Concurrently, parts G5(H-8-6) and G5(F-10-7) have also merged. Also, the final scheduling time of DRUM is 3,216 minutes, thereby saving 120 minutes. Figure 9f summarizes the merged results.

Stage 4: Hybrid Scheduling

Initially, we schedule prior operations of the core operation M4 (operations E5, E5, G5, G5, C5, C5, G5, E5, E5, C5, G5, G5, E5 in sequence) backwards from its scheduled starting time. For instance, the prior operation of part E5(H-12-2) is null; therefore, we skip to part E5(F-10-3). Also, the prior operations E1, and E3 of part E5(F-10-3) are scheduled backwards close to core operation M4 of part E5(F-10-3). Second, we schedule subsequent operations of the core operation M4 (operations E5, E5, G5, G5, C5, C5, G5, E5, E5, C5, G5, G5, and E5 in sequence) forwards from its scheduled finish time. For instance, the subsequent operations F7, H8, and H9 of part G5(H-12-2) are scheduled forwards close to core operation M4 of part G5(H-12-2). Third, we schedule the free orders (parts B-10-1, D-15-2, D-14-4, D-12-5, and D-12-7 in sequence) backwards from its due-date. Fourth, we calculate the part's minimum-in-processing time. For instance, part H-12-2's minimum-in-processing time is equal to 1,346 (1,346-0) minutes. By continuing the step, Figure 10 and Table 3 display the scheduled results.

4. Conclusions

This study has developed a novel scheduling heuristic algorithm for job shop manufacturing environments. Also, previous research has not focused explicitly on the job's due-date performance; thus, setup times may have been reduced at the expense of due-date performance. In practice, firms consider meeting the due-date as the most important performance criterion. Nevertheless, the proposed novel scheduling heuristic for job shop manufacturing system based on Capacity Constraint Resource (CCR) can accurately reflect the realities of the production environments by stressing good due-date performance, while reducing overall setup times. The proposed algorithm's strong performance indicates that our four-stage scheduling heuristic, with its more sensitive switching procedures, can not only overcome some of the weaknesses shown by DBR scheduling heuristics, but can also glean setup time savings without sacrificing due-date performance. For environments that are conducive to job shop scheduling, the results of this study provide a valuable reference for a firm attempting to develop the appropriate scheduling heuristics.
Figure 6: The Flowchart of Backward and Forward Scheduling Mechanism
Table 1: Product's Demand Quantity and Due-Date

<table>
<thead>
<tr>
<th>B Product</th>
<th>Quantity</th>
<th>Due-Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>D Product</td>
<td>Quantity</td>
<td>Due-Date</td>
</tr>
<tr>
<td>F Product</td>
<td>Quantity</td>
<td>Due-Date</td>
</tr>
<tr>
<td>H Product</td>
<td>Quantity</td>
<td>Due-Date</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B Product</th>
<th>Quantity</th>
<th>Due-Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>D Product</td>
<td>Quantity</td>
<td>Due-Date</td>
</tr>
<tr>
<td>F Product</td>
<td>Quantity</td>
<td>Due-Date</td>
</tr>
<tr>
<td>H Product</td>
<td>Quantity</td>
<td>Due-Date</td>
</tr>
</tbody>
</table>

Table 2: Workcenter Data

<table>
<thead>
<tr>
<th>Machine</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
<th>M6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Setup Time (Minutes)</td>
<td>60</td>
<td>30</td>
<td>60</td>
<td>60</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 7: Product's Routing Layout and Work-In-Process
Figure 8: The Detailed Analysis of Product's Routing
A Scheduling Heuristic for Job Shop Manufacturing System

Figure 9a: Create RUIN

Figure 9b: Level RUIN

Figure 9c: Create DRUM

Figure 9d: Setup Time Savings for DRUM(I)

Figure 9e: Setup Time Savings for DRUM(II)

Figure 9f: Setup Time Savings for DRUM(III)
Figure 10: The Results of Hybrid Scheduling

Table 3: The Product's Finish Time, Release Time and Flowtime

<table>
<thead>
<tr>
<th>Product</th>
<th>Due-Date ($d_i$)</th>
<th>Finish Time ($c_i$)</th>
<th>Lateness ($c_i - d_i$)</th>
<th>Release Time ($r_i$)</th>
<th>Flowtime ($c_i - r_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-10-1</td>
<td>480</td>
<td>480</td>
<td>0</td>
<td>170</td>
<td>310</td>
</tr>
<tr>
<td>D-15-2</td>
<td>960</td>
<td>960</td>
<td>0</td>
<td>370</td>
<td>590</td>
</tr>
<tr>
<td>H-12-2</td>
<td>960</td>
<td>1,346</td>
<td>386</td>
<td>0</td>
<td>1,346</td>
</tr>
<tr>
<td>B-09-3</td>
<td>1,440</td>
<td>1,651</td>
<td>211</td>
<td>1,192</td>
<td>459</td>
</tr>
<tr>
<td>F-10-3</td>
<td>1,440</td>
<td>1,432</td>
<td>-8</td>
<td>162</td>
<td>1,270</td>
</tr>
<tr>
<td>D-14-4</td>
<td>1,920</td>
<td>1,828</td>
<td>-92</td>
<td>1,046</td>
<td>782</td>
</tr>
<tr>
<td>B-16-4</td>
<td>1,920</td>
<td>2,100</td>
<td>180</td>
<td>1,007</td>
<td>1,093</td>
</tr>
<tr>
<td>D-12-5</td>
<td>2,400</td>
<td>2,292</td>
<td>-108</td>
<td>1,546</td>
<td>746</td>
</tr>
<tr>
<td>H-08-5</td>
<td>2,400</td>
<td>2,388</td>
<td>-12</td>
<td>1,476</td>
<td>912</td>
</tr>
<tr>
<td>B-11-6</td>
<td>2,880</td>
<td>2,865</td>
<td>-15</td>
<td>2,079</td>
<td>786</td>
</tr>
<tr>
<td>H-08-6</td>
<td>2,880</td>
<td>3,064</td>
<td>184</td>
<td>1,848</td>
<td>1,216</td>
</tr>
<tr>
<td>D-12-7</td>
<td>3,360</td>
<td>3,360</td>
<td>0</td>
<td>2,652</td>
<td>708</td>
</tr>
<tr>
<td>F-10-7</td>
<td>3,360</td>
<td>3,446</td>
<td>86</td>
<td>2,516</td>
<td>930</td>
</tr>
</tbody>
</table>

Average(Batch): 62.46, 857.53

Note: Total Tardiness = 1,047 Minutes, Total Earliness = 235 Minutes.
References


Goldratt, E.M., 1990b, "What is this thing called theory of constraint and how should it be implemented," North River Press, Inc.

