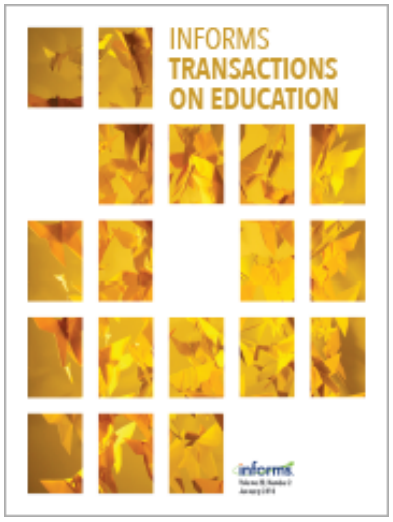


INFORMS Transactions on Education

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Interactive Excel-Based Gantt Chart Schedule Builder

Sarah G. Nurre, Jeffery D. Weir

To cite this article:
Sarah G. Nurre, Jeffery D. Weir (2017) Interactive Excel-Based Gantt Chart Schedule Builder. INFORMS Transactions on Education 17(2):49-57. https://doi.org/10.1287/ited.2016.0168

http://pubsonline.informs.org/journal/ited/

INFORMS TRANSACTIONS ON EDUCATION
Vol. 17, No. 2, January 2017, pp. 49–57
ISSN 1532-0545 (online)

Interactive Excel-Based Gantt Chart Schedule Builder

Sarah G. Nurre,[a] **Jeffery D. Weir** [b]

[a] Department of Industrial Engineering, University of Arkansas, Fayetteville, Arkansas 72701; [b] Department of Operational Sciences, Air Force Institute of Technology, WPAFB, Ohio 45433

Contact: snurre@uark.edu (SGN); jweir@afit.edu (JDW)

Received: September 26, 2014
Accepted: July 22, 2016
Published Online: January 13, 2017

https://doi.org/10.1287/ited.2016.0168

Abstract. Many scheduling dispatching rules are intuitive processes used in every day life. For example, when faced with a variety of tasks due at different times one often implements the earliest due date scheduling rule: The next task worked on is the one with the earliest due date. Other common scheduling dispatching rules are easily understood, thereby enabling one to devise the rule when given the opportunity to experiment via trial and error. In this paper, we present an interactive Excel-based Gantt Chart Schedule builder that enables students to experiment with building schedules for different single and parallel machine problem examples. Instead of explicitly telling students these common scheduling rules, the schedule builder enables students to gain intuition about the rules on their own. Herein we describe the interactive schedule builder we created, explain how instructors and students can use this tool, perform a small preliminary assessment on the student perception metric, and provide supplemental teaching materials enabling use of the schedule builder in a variety of classroom environments.

Supplemental Material: Supplemental material is available at https://doi.org/10.1287/ited.2016.0168.

Keywords: Gantt Chart • scheduling • excel • interactive tool

1. Introduction

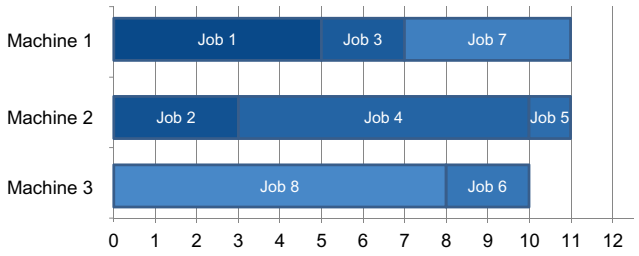
Many of the state-of-the-art scheduling rules and heuristics, often called dispatching rules, are concepts adopted in every day life. When we ask our students how they determine which of their homework assignments they work on next, the most frequent answer is "*The one due next*." This decision making procedure is called the earliest due date (EDD) rule, which is optimal for the scheduling problem with one machine (e.g., one student) and the objective of minimizing the maximum lateness (e.g., most late homework assignment) denoted by $1||L_{\max}$ (see Pinedo 2012). The EDD dispatching rule is optimal for $1||L_{\max}$. Even better, it is easily understood and intuitive for a naïve decision maker: You work on the task with the next earliest due date. There is a portfolio of dispatching rules with these properties, which can be concisely conveyed as follows: Next, schedule the job that (i) gives you the best weight for time ratio, i.e., weighted shortest processing time (WSPT), (ii) has the smallest processing time (SPT), (iii) has the longest processing time (LPT), and (iv) precedes the largest number of future tasks, i.e., largest number of successors (LNS). While not all concise scheduling dispatching rules are optimal, they do have good approximation guarantees.

For an instructor these rules are desirable. A student provided with the appropriate foundational knowledge should be able to experiment and build intuition about these dispatching rules. If the student does not arrive at the rule exactly, he or she should be able to use trial and error to see the properties of high performing schedules. To enable this process of trial and error experimentation, we developed an Excel-based interactive Gantt Chart Schedule Builder.[1] The schedule builder is designed to complement traditional scheduling course lectures, thus enabling students to build Gantt Charts (see Figure 1). Instead of being taught a subset of the common scheduling dispatching rules, students can discover and build intuition about the problem, and identify the properties on their own.

The schedule builder first allows a user to input a specific scheduling problem example with a set number of machines, jobs, parameter values (e.g., processing times, due dates), and constraints (e.g., precedence, preemptions). Then the user can build a schedule and click a button to measure the quality of that schedule using objective functions consistent with the classic scheduling book (Pinedo 2012). After seeing the quality of the schedule, the user can interactively adjust which jobs are assigned to which machines and the corresponding sequences.

We believe this experimental process enables students to build intuition about creating schedules and confidence when they observe key properties. Additionally, the schedule builder compliments traditional lectures directed toward auditory learners by enabling visual learners to see a specific schedule, and kinesthetic learners to touch and move around the jobs

Figure 1. Example of a Gantt Chart Which the Schedule Builder Enables a User to Create



on different machines (see Barbe et al. 1979). We also included added functionality in the schedule builder. Given access, this functionality algorithmically builds a schedule based on a selected scheduling rule. This functionality can be revealed to students after they have successfully experimented with scheduling problem instances to discover properties and rules for building high quality schedules.

The schedule builder was first used in the Summer of 2014 in a three-credit (10 week) course called OPER 626/LOGM 631 Scheduling Theory. This course is cross-listed in the Operational Sciences and Logistics Management departments. Although the intended audience is Master's and Doctoral students pursuing degrees in Operations Research and Logistics Management, enrollment is open to all graduate students. Most often, students from outside these two departments are in Systems Engineering. The schedule builder was introduced on the first day of class via a demonstration by the instructor of the tool's functionality. The schedule builder focuses on deterministic scheduling problems in a single or parallel machine environment.

In the first homework assignment, we asked the students four similar questions. In the first part of each question, students were asked to use the tool to find the optimal schedule for a specific scheduling problem instance. Second, we asked the students to experiment with other instances of the stated problem (i.e., the same machine environment, objective, and constraints but a different number of jobs and parameters) and to note the properties present across the optimal or high performance schedules. In the third and fourth parts of each question, students were asked to devise a generic rule for solving instances of the scheduling problem. The intent of each question was to encourage experimentation via trial and error, enabled with the schedule builder, and to observe properties of high performance schedules.

Students were asked to provide feedback on the schedule builder after this first homework assignment and throughout the duration of the course. Based on the feedback, the schedule builder went through thirteen iterations. Throughout the rest of the course, the students were not explicitly asked to use the tool, however the tool was used for in-class examples. Additionally, because we created the tool in VBA within Excel, the students were able to use it at home and in school labs.

We performed a preliminary assessment of the tool based on the student perception metric (see Kane 2012). At the end of the course, students received a questionnaire covering the following main topic areas: (i) how they used the tool and the frequency of use, (ii) their perception of the usefulness of the tool, and (iii) ideas for improvement. Results and analysis of the answers from this questionnaire are provided in Section 3. As this is a preliminary assessment, we do not draw major conclusions about the tool's effectiveness; instead, we demonstrate how the students in our class perceived the schedule builder. It is our hope that by providing this preliminary assessment and showing that our students found value in the tool, other instructors will see the benefit of adopting this tool in their classrooms. Future researchers should conduct a more thorough assessment and capture many performance metrics via different evaluation methods.

The idea of using electronic tools as teaching aids is not new. For operations research and management science courses, others have used tools to aid in course instruction, covering topics in inventory modeling (see Liu et al. 2013), queueing (see Leong 2007), decision making under uncertainty (see Chan 2013), statistics, and operations management. Mason (2013) developed SolverStudio, which easily integrates algebraic modeling language for mathematical programming in an Excel environment. A valuable feature of this is that it is an Excel tool thereby catering to the many users who are comfortable in this environment. Scheubrein and Kulturel-Konak (2006) also maintain the spreadsheet environment by discussing a teaching tool for an MBA operations management course. Specific tips for implementing interactive tools in an operations management course are discussed by Snider and Balakrishnan (2013). For statistics courses, instructors are referred to Enns (2008) for design of experiments, Balakrishnan and Oh (2005) for statistical process control, and Tsai and Wardell (2006) for business statistics.

Furthermore, others have used tools and specific instructional methods for teaching scheduling and project management courses. Baker (2005) created a spreadsheet-based tool and Sniedovich (2005) presented a specific teaching procedure for the critical path method. A spreadsheet-based tool for implementing dynamic programming (DP) algorithms is presented by Raffensperger and Richard (2005). While this tool is not solely for scheduling, DP algorithms are a common methodological approach for solving scheduling problems; one of the examples detailing the tool is a scheduling problem.

There are a variety of commercial software packages with Gantt Chart features (e.g., Smartsheet 2015, TeamWeek 2014, and AIMMS 2015). Furthermore, included with Pinedo (2012), a common book used for teaching a course in scheduling, are three software systems: Lekin, LiSA, and TORSCHE.

Lekin (see Pinedo et al. 2002) is an older system that only runs on Windows 95, 98, and NT. While we were unable to get the software to work due to lack of compatibility with our more up to date operating systems, this scheduling system appears to be the most similar to the scheduling tool presented here. However, even if a user had access to older operating systems, the book states that the free educational version of the tool included with purchase of Pinedo (2012) only includes a teaching tool for job shop scheduling, whereas we consider the single and parallel machine scheduling environments.

The library of Scheduling Algorithms (LiSA) (see Andresen et al. 2015) is a software package that can solve deterministic scheduling problems. In this system a user inputs data associated with a scheduling problem and then selects and invokes an algorithm from the library of choices. An output of this action is a Gantt Chart visually displaying the assignment of jobs to machines and the sequence in time. TORSCHE (see Šůcha et al. 2006) is a system targeted towards researchers to aid in development of complex scheduling algorithms. An output of this tool is also a Gantt Chart. While both of these tools fill a niche in the scheduling field and visually display a Gantt Chart, the software packages are not designed for teaching and do not allow the user to interactively *create* schedules to build intuition about classic scheduling rules on their own through trial and error placement of jobs to machines. Instead, these tools are more sophisticated and include built-in algorithms that can be used in conjunction with cutting edge new algorithms.

The schedule builder presented here embraces the idea of enhanced discovery learning (see Manzano 2011). With enhanced discovery learning, students are provided with sufficient knowledge as a stepping stone between basic concepts and those which an instructor wants students to discover . Although a full assessment should be conducted to determine whether this tool creates an enhanced discovery learning environment, by allowing students to actively create schedules and build intuition about concepts, we embrace the general idea of active discovery. This idea is not new. Others have considered it when teaching operations research and management science courses. Kydd (2012) used an interactive applet to teach the basic and visual aspects of linear programming. The applet enables active learning by allowing students and teachers to see the impacts of adjusting components of a linear program such as the iso-profit line.

Main Contributions. The main contributions of this paper are as follows: (i) creation of an Excel-based interactive Gantt Chart Schedule builder available for use by instructors and students, (ii) a thorough explanation of how instructors and students can use the tool inside and outside the classroom, (iii) a preliminary analysis of how students use and perceive the benefits of the scheduling tool, and (iv) dissemination of associated teaching materials that can be combined with the schedule builder to help students build intuition about properties of high performance schedules.

The remainder of this paper organized as follows: In Section 2, we provide a detailed description of the schedule builder and its many functions. In Section 3, we present the results of a preliminary assessment indicating how our students perceived the schedule builder and provide suggestions for its use in a course. We conclude in Section 4 with a summary of the main contributions and proposals for future applications.

2. Description of the Tool

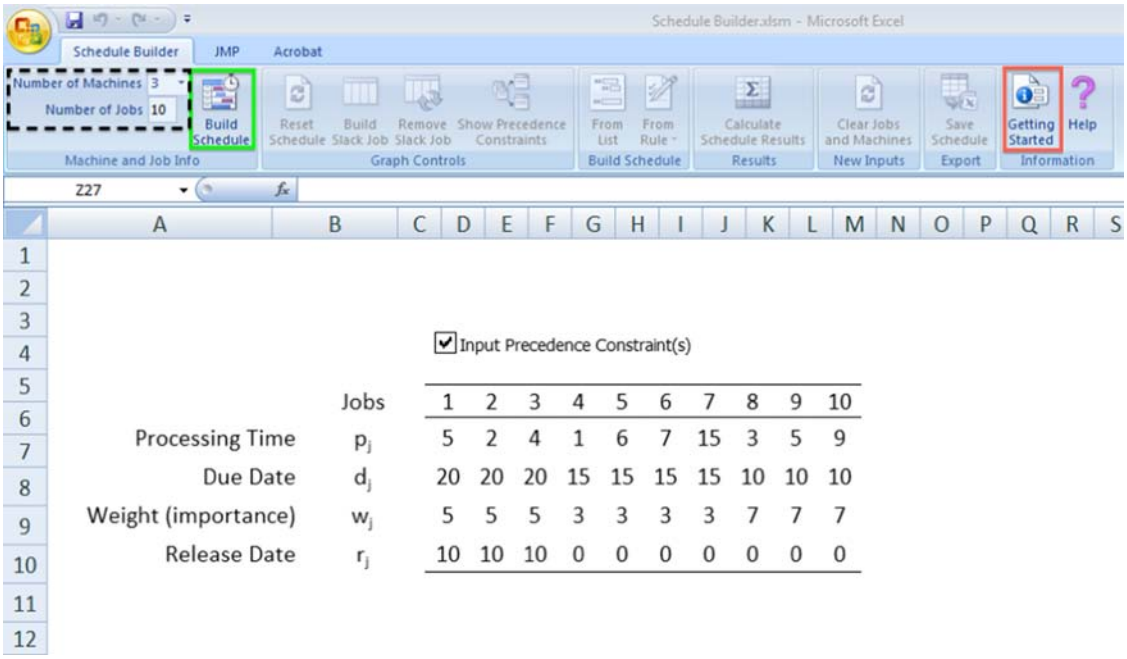
In this section, we describe the schedule builder tool. We include the necessary input data, available functions, opportunities for user trial and error when building a schedule, security features preserving the accuracy of the tool, and error checking on the schedule. We developed the schedule builder in VBA with Excel. Thus the computing requirements for running the tool are Excel 2007, 2010, 2013 or 2016 for Microsoft Windows operating systems.

2.1. Input Data

When opening the schedule builder, the user should first enable macros. Two splash screens will appear. The first indicates the name of the tool and contact information, and the second provides a high level overview of the purpose of the tool and basic instructions for getting started. Users may click a check box if they do not need to see the second splash screen when opening the tool in the future. However, users may get the information from this second splash screen by clicking on the *Getting Started* button on the file menu ribbon specific to the schedule builder (see Figure 2 red box).

To input a scheduling instance, a user should first input the number of machines and jobs in the space indicated on the ribbon (see Figure 2 black dashed box). The schedule builder can accommodate up to four parallel identical machines and up to 22 jobs. Next, a user should input the parameters specific to the scheduling instance where, at a minimum, the processing times p_j for each job j are required and optional inputs include due date d_j, weight w_j, and release date r_j. All values input must be 0 or a positive integer. Precedence constraints stipulating that job j must be completed before job k may begin processing are allowed when the

Figure 2. Screenshot of the Main Menu with the Input Data for Example 1

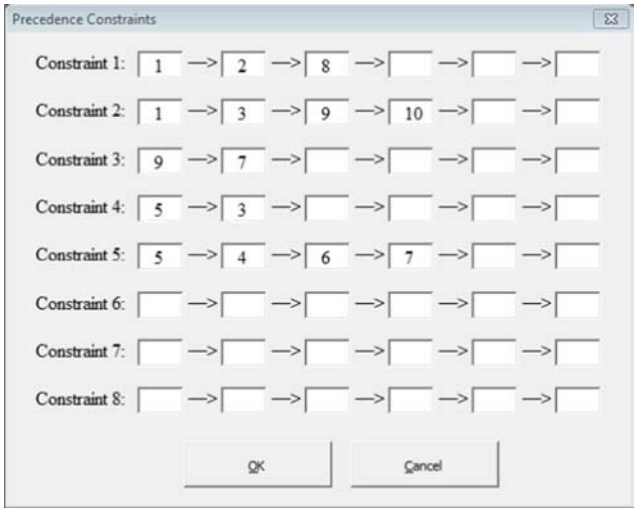


Note. The *Getting Started* button is outlined in red, Machine and Job number selection in dashed black, and the *Build Schedule* button in green.

precedence check box is selected. In Figure 2, we show the input for Example 1, which will be used throughout Section 2. In this example, we consider three parallel machines, 10 jobs, precedence constraints, and the parameter values shown in Figure 2. After the user inputs all data, she should select the *Build Schedule* button (see Figure 2 green box).

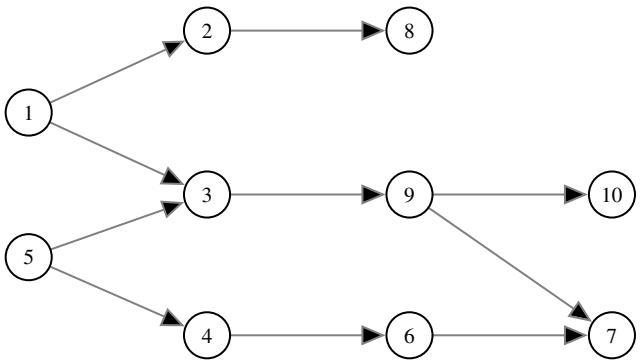
If the box indicating that precedence constraints are present is checked, a pop-up will appear after the user selects the *Build Schedule* button. As shown in Figure 3, the precedence constraints must be entered as chains, however all traditional precedence constraints can be represented by a series of chains. For example, in the simplest case, a user can enter each precedence constraint as an individual chain in which job i must precede job j. Thus, through a minor transformation, any set of traditional precedence constraints can be represented as a chain. Longer chains are also allowed. In Figure 4, we present a precedence graph for Example 1 in which each job is represented as a node and the arrows between the nodes represent the precedence relationships. For example, from this graph we

Figure 3. The Pop-Up That Appears After the *Build Schedule* Button is Pressed if the Box Indicating Precedence is Checked



Note. In this pop-up the user can input the precedence relationships and select *OK*, otherwise select *Cancel*.

Figure 4. Precedence Graph for Example 1 in Which Each Job is Represented as a Node and the Precedence is Represented by Arcs



Note. For example, jobs 1 and 5 must be completed before starting job 3.

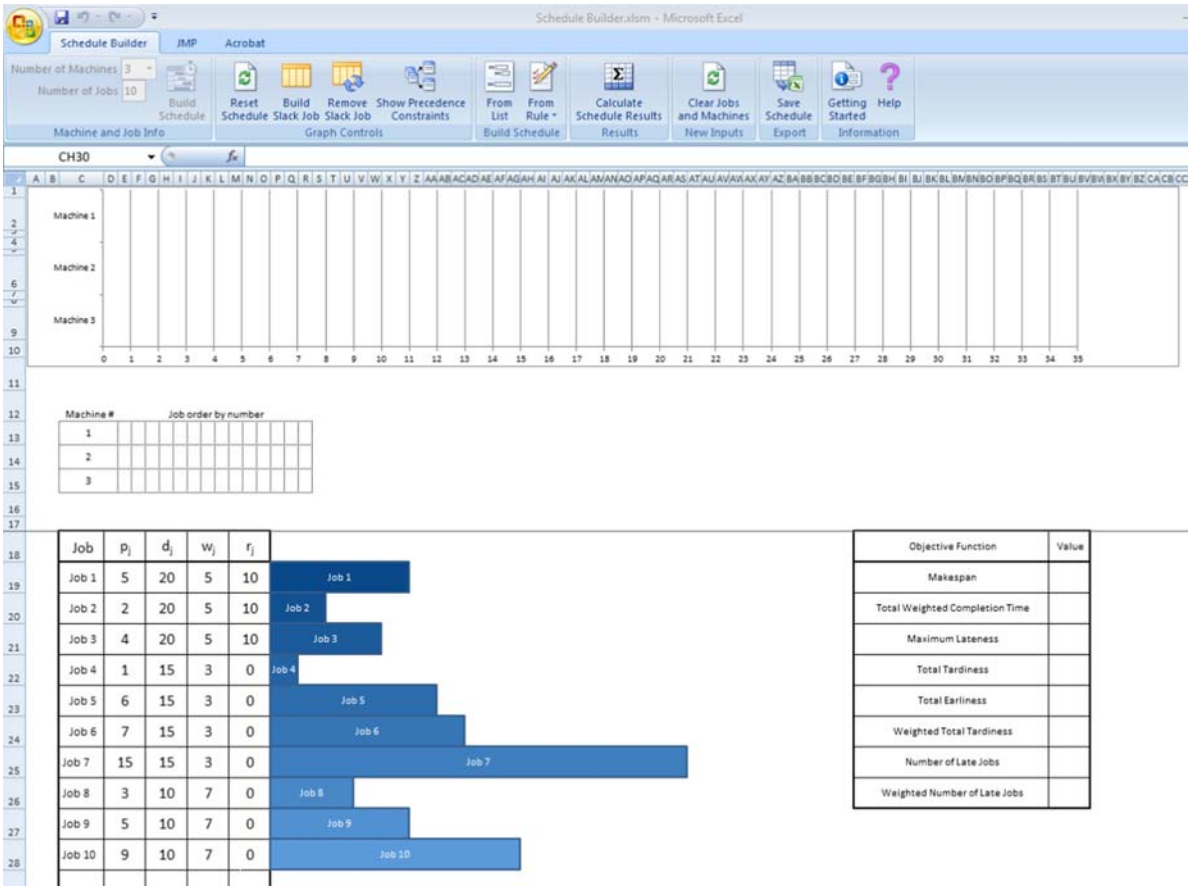
can conclude that job 5 must be complete before job 4 begins, and also jobs 6 and 9 must be complete before job 7 begins. We translated this traditional precedence graph into the following series of chains: $1 \rightarrow 2 \rightarrow 8$, $1 \rightarrow 3 \rightarrow 9 \rightarrow 10$, $9 \rightarrow 7$, $5 \rightarrow 3$, $5 \rightarrow 4 \rightarrow 6 \rightarrow 7$. A user can input these chains into the pop-up as shown in Figure 3. Note that this is not the only way to enter this set of precedence constraints as chains, but merely one possible example. All possible chains (including individual chains) are valid inputs for this pop-up box.

2.2. Interactive Gantt Chart Functionality

After the user selects the *Build Schedule* button and enters precedence constraints (if applicable), the tool takes the user to the interactive Gantt Chart maker (see Figure 5). Across the top is the Gantt Chart with a time horizon out to 35 time units and rows for the number of machines selected. Below this is space enabling the user to build a schedule from a list. At the bottom left is the list of jobs, the associated job parameters, and blue rectangles scaled to the size of each job's processing time. Last, at the bottom right is a box that displays the objective function value for common scheduling objectives based on the currently built schedule.

There are three ways to build a schedule from the input instance. Option one is to "drag and drop" the blue rectangles for a job and place them on the Gantt Chart. When placing each job, it is best to position the rectangle slightly to the right and below the desired location, at which point the rectangle will snap to a grid. Each blue rectangle representing a job can be moved around on the Gantt Chart until the desired schedule has been completed. The second option for building a schedule is from a list. Directly below the Gantt Chart, the user can type in the job number (e.g., 7 for job 7) in the row of the machine for which it should be assigned and in the desired sequence in relationship to the other jobs. The user must then select the *From List* button in the top ribbon, which places the jobs accordingly. At this point, if a change is sought in the schedule the list can be updated (and the button selected again) or the job rectangles can be dragged and dropped to new locations. The last option for building a schedule is from a common scheduling dispatching rule. When pressed, the *From Rule* button expands to a drop down list where one rule from the SPT, WSPT, LPT, EDD, and LNS rules can be selected at which time the selected rule is executed for scheduling the jobs.

Figure 5. Interactive Gantt Chart Portion of the Tool, with the Gantt Chart at the Top, Schedule Building via a Typed List Below, Jobs with Associated Parameters at the Bottom, and Objective Function Calculations at the Bottom Right



Note that this *From Rule* functionality can be hidden for the case when students are looking for common scheduling dispatching rules.

Placement of the full rectangles coincides with scheduling jobs in a nonpreemptive environment where once a job begins processing it must continue until it is complete. We included functionality that allows for the preemptive environment by allowing the processing of a job to be interrupted. In this environment, a job is finished when the sum of its processing over all machines equals its total processing time. This interruption of jobs can be equivalently captured by scheduling portions of a job in smaller segments or subjobs (e.g., a job j with $p_j = 5$ could have smaller subjobs with processing equal to 1 and 4 or 2 and 3). In the schedule builder, if a user clicks on the job name in the *Job* column a pop-up box will appear asking "How many segments do you want Job j broken into?" Acceptable inputs into the box include integer values at least equal to 1 and no greater than the processing time of job j. A pop-up will next appear asking for the processing time of each segment, where the smallest processing time allowed is 1 time unit. After this process, the rectangle to the right of the job will be broken into s rectangles, where s is the number of segments input into the pop-up box. Furthermore, each rectangle is scaled according to the processing time entered. These job rectangles maintain the same functionality as the other job rectangles as they can be "dragged and dropped" or entered into the list when called by their proper name. The name of each job segment for job j is now denoted by $j.1, j.2, \ldots, j.s$ if $s > 1$, otherwise by j if $s = 1$. At any point, if the user wants to adjust the number of segments or processing times associated with each segment, this series of steps can be repeated. Note that, consistent with the scheduling definition of preemption, the user should be careful not to schedule any portion of a job for processing on two machines at one time (i.e., if job j is broken into $j.1$ and $j.2$, $j.1$ and $j.2$ cannot be processed during any time t).

Delays or idle time are required (or beneficial) for some schedules. We revisit Example 1 to demonstrate such a schedule. In Example 1, there are three machines available, however, based on the precedence constraints displayed in Figure 4 only two jobs, 1 and 5, are available for processing at time 0. Furthermore, when we examine the release dates, job 1 is not available for processing until time 10. This results in idle time on machines 2 and 3 for at least p_5 time periods. To account for this idle time and other instances where delays are desired, white space can be left on the schedule or slack/"dummy" jobs can be introduced. Using the schedule builder, a slack job can be introduced in two ways. The first is by clicking the *Build Slack Job* button, which results in a pop-up where the processing time for this slack job should be entered. The other option is to enter Si in the Build Schedule by List section, for slack job i (e.g., enter $S1$ when introducing the first slack job). Then when clicking the *From List* button, a pop-up will appear for each slack job Si asking for the processing time associated with each job. By contrast to a real job, a slack job appears in red for ease of differentiation. Note again that slack jobs are not required any time a delay occurs; instead a delay can occur by not scheduling a job on a certain machine at specific times, thus resulting in white space. However, slack jobs are available to explicitly input a delay and are not considered in the objective function calculations. Furthermore, a slack job can be removed by selecting the *Remove Slack Job Button*.

The other functionalities of the tool can be triggered via the buttons in the top ribbon. The *Reset Schedule* button moves all jobs currently placed on the Gantt Chart down to their original starting position. The *Show Precedence Constraints* button allows a user to see and adjust the current set of constraints. If all jobs have been placed on the Gantt Chart and satisfy the constraint set (i.e., no job is scheduled before its release date and precedence constraints are satisfied), the *Calculate Schedule Results* button calculates the set of objectives and displays the values in the bottom right of the tool. For information about the meaning and calculation of these objectives, see Pinedo (2012). The *Clear Jobs and Machines* button resets the problem instance, taking the user back to the main menu. Last, the *Help* button displays a pop-up box providing detailed descriptions of the schedule builder functionality. When a user is satisfied with a schedule created, it can be saved by clicking the *Save Schedule* button where options to save the schedule to pdf, a new Excel workbook or an existing Excel workbook are offered. Note that to save a schedule to a new or existing workbook the user must have a check box selected trusting VBA project objects. If a user does not currently have this check box selected a series of instructions are shown that walk the user through the steps necessary to trust a VBA project object.

2.3. Security and Error Checking

In this subsection, we discuss the error checking in place to ensure that a built schedule is feasible and the security features of the tool protecting its correct use. Error checking the feasibility of a built schedule occurs when the user presses the *Calculate Schedule Results* button. A pop-up indicating an error occurs if the schedule has any of the following four characteristics: (i) A job does not satisfy the precedence constraints (i.e., if in the precedence constraints job i proceeds job j, an error will occur if job j is scheduled to start before completion of job i); (ii) A job violates its release date (i.e., if job j is scheduled to start at a time $t < r_j$); (iii) Two jobs overlap on a machine, indicating that a machine is

working on more than one job at a time; (iv) At least one job is not scheduled. With this last characteristic, we require that all jobs, including all segments of jobs broken up for preemption and all created slack jobs, be placed on the Gantt Chart for the results of a schedule to be calculated. If slack jobs are no longer required, the user should select the *Remove Slack Job* button to eliminate unnecessary slack jobs.

To ensure the integrity of the tool, we input security features so that users do not accidentally inhibit the proper calculation and features of the tool. In one version of the tool, the VBA code has been password protected. For interested users, we also provide a non-password protected version of the tool to allow for future enhancements and customizations of the VBA code. As another security measure, we lock all cells that do not require user input. The locked cells include all cells on the interactive Gantt Chart tab except the build a schedule by list and job parameter cells. The job parameter cells can be adjusted if the user wants to alter the processing time, weight, release date or due date of any job. Contact information is provided in the *Help* section if a user finds bugs or has suggestions for improvements.

3. Use and Assessment

In this section, we discuss ways in which we envision use of the schedule builder during a course. We also present the results of a small preliminary assessment capturing the student perception metric (see Kane 2012).

We used the schedule builder in a Scheduling Theory course where students came from a variety of backgrounds. In the Summer 2014 course, there were two Master's Operations Research students, one Master's Logistics student, one Master's Systems Engineering student, and one Ph.D. Operations Research student. We believe the schedule builder helps make differing backgrounds less of an issue as students can visualize the concepts introduced and more readily participate in classroom discussions. While the schedule builder was only used in a course with graduate students (due to the authors' institution), we believe the tool could be easily used in an undergraduate scheduling course or even in a lecture (or lecture sequence) in a course such as Deterministic Operations Research.

In this paper, we provide a supplementary assignment that instructors can use to help facilitate students' use of the tool with hopes that they build intuition about schedules. The assignment is similar to the one used in the Summer 2014 course with modifications based on observations and feedback. The assignment walks students through four scheduling problems asking them to first find the optimal solution, then observe properties of the solution, experiment with other instances of the same problem, and devise a rule for solving instances of this scheduling problem based on the properties observed in the optimal and high performance schedules covered during the experimentation phase. Before giving this assignment, we encourage instructors to provide students with enough foundational knowledge about scheduling and rules to solve scheduling problems. We hope that this foundational knowledge will help students feel comfortable with the task and experiment through trial and error to observe key concepts. By foundational knowledge, we mean that, at a minimum, students are exposed to scheduling terminology as in the $\alpha \mid \beta \mid \gamma$ notation with possible inputs for each of the triplet components, and introduced to the schedule builder in class where the instructor covers a full example showing how trial and error can be used to see those schedules that lead to better objective function values. Note that all scheduling rules that students create will not be optimal as there are many NP-hard scheduling problems (see Pinedo 2012). However, we think it is beneficial for students to make observations about the properties of many high performing schedules. If a student devises a rule and then determines that this rule does not produce the optimal solution for a specific instance, we believe she is learning that not all dispatching rules produce the optimal solution, and further that there are some problems that do not have a polynomial time rule that always produces the optimal solution.

We collected informal feedback throughout the quarter, including suggestions for improvement of the tool. In the last week of the course, we formally solicited feedback from the students via an anonymous questionnaire. The questions address how and how frequently students used the tool, their perception of the tool's effectiveness, and suggestions for improvement. By the time students were given this questionnaire, they had experience with all versions of the tool. The changes made, from the initial tool to the final version, were directly based on informal student feedback received throughout the quarter. In the questionnaire, we posed the following questions:

1. How frequently did you use the scheduling tool? Please indicate at which portions of the quarter.
2. What did you use the scheduling tool for? Did you just use it for Homework 1 or did you also use it at other times (e.g., for other homework, examples from the book, . . .)?
3. Do you believe the scheduling tool was useful for understanding the scheduling concepts? Why or why not?
4. Did the scheduling tool enable you to more easily use trial and error to visualize and determine properties of schedules that are good for different objectives?
5. Is the scheduling tool easy to use or did you experience difficulties?

6. Do you have suggestions for improvement regarding the scheduling tool?

7. Do you have suggestions for improvement regarding use of the scheduling tool with this course (e.g., use for more in-class examples)?

All five students in the course returned the questionnaire. As this is a small number, future research should fully assess the student perception and other performance metrics. However, in sharing this feedback it is our intention to indicate how our students perceived and used the tool, and thus to inform others of the benefits of the schedule builder.

In response to questions 1 and 2, all students indicated that they used the tool during the first two weeks of the course. Three of the five students used it at other times throughout the quarter, for other homework assignments or to prepare for exams. This response coincides with our intention for the tool. Specifically, we believe the visual environment that the tool provides facilitates expanding intuition about concepts but is not as beneficial when delving into harder constructs such as complexity and algorithm approximations, which are presented later in the course. Thus, we were pleased with this use of the tool for the beginning of the course and gratified that some students saw the benefits of the tool and continued to use it throughout the quarter.

In response to question 3, all students indicated that the tool was helpful and improved their understanding of scheduling concepts. One student indicated that "... [the schedule builder] allows you to easily check all permutations for a particular setting." Another student liked the tool for verifying his understanding of the different scheduling problems and how objectives are calculated. Even with these favorable responses, one student indicated that "...it was useful at the beginning, however manually building the gantt charts [on graph paper] was a better exercise for me to help solidify [a] lesson." Thus, while all students indicated that the tool was useful, at least one preferred pencil and graph paper. We asked question 4 to learn if the students believed that the tool enabled them to easily determine properties of high performance schedules for different objectives. One student said "Absolutely"; all other students responded "yes" (one student underlined this word twice). Thus, we conclude for this small sample that the scheduling tool enabled us to achieve our desired result: Through experimentation of assigning and sequencing jobs on machines enabled by the schedule builder, students easily discovered properties of high performance schedules.

In question 5, we wanted to learn about the students' perception of the usability of the tool. Most students indicated that the tool improved with each version and that the last version "seems pretty nice." All students agreed that the tool was "very easy to use" or "very intuitive." In terms of feedback, some students said that when dragging and dropping jobs, the tool was sometimes sluggish. As noted in Section 2.2, we recommend that users place jobs slightly to the right and below the desired location; this enables the jobs to snap to the grid. In question 6, we sought suggestions for improving the tool. One student suggested that a feature be added whereby the user inputs the Graham notation for a scheduling problem (i.e., $\alpha \mid \beta \mid \gamma$, see Pinedo 2012), and that the tool would then suggest a scheduling rule. While we appreciate this suggestion, such a feature would eliminate the experimental aspect of using trial and error to learn about schedule properties. One valuable suggestion was to include an option to automatically enumerate all schedules. Although we think this feature would help students to visually see the difference in all permutations, we did not introduce it because we want to expand students' intuition when thinking about a specific scheduling problem. They would then use the tool to first easily confirm or deny this intuition, and second to make observations about high performance schedules. Another student suggested that a feature be introduced to save specific schedules. We updated the tool to include the ability to save a specific schedule to pdf, a new Excel workbook or an existing Excel workbook.

In the last question, we asked for suggestions about classroom use of the tool. The responses were split. Three students said that the current use of the tool in the classroom was appropriate; two suggested that the tool be used more frequently in class. Overall, we were very pleased with the feedback received. Ultimately, we believe that for this small set of students the tool provided value inside and outside the classroom. Also, at a minimum, students were exposed to the process of trying to develop scheduling rules and methods on their own. Most real-world scheduling problems are complex and do not always have an easy-to-describe rule; complex scheduling problems sometimes require development of complex scheduling rules. Thus, while more assessment should be performed, we hypothesize that because the students experienced developing scheduling rules based on observed properties through the schedule builder, they will feel more comfortable and experienced when developing more complex scheduling rules as they encounter real-world scheduling problems.

4. Conclusions

In this paper, we described an Excel based interactive Gantt Chart Schedule Builder. We created the schedule builder so that it can be used in a scheduling course or as part of a scheduling module in other courses such as an operations research course. We described

the features of the schedule builder, including the various inputs allowed, user features such as allowing preemption or precedence constraints, and output of the schedule including the visual Gantt Chart and calculation of traditional scheduling objectives. We further describe how we used this tool in a scheduling course to allow students to use trial and error to build intuition about problems and to observe properties about high performance schedules. Based on these observations and the intuition developed, we asked students to devise a rule for assigning and sequencing jobs on machines. We performed a small preliminary assessment of the tool to determine students' perception metric. Based on the feedback, we believe this tool is an effective supplement to traditional classroom instruction and that it helps visual and kinesthetic learners. The tool only considers single and parallel machine scheduling environments. Future research should consider expanding to include shop environments such as flow shop, job shop, and open shop. Other scheduling problem specifics should also be considered for inclusion such as sequence dependent set up times, batch processing, machine speed, and storage on machines. We built the tool in VBA and thoroughly tested it only on Windows operating systems. Further testing and development should be conducted for building a tool that is compatible with Mac, Linux, and Unix. Last, an in depth assessment should be conducted to quantify the effectiveness of the tool. With this assessment, it would be interesting to compare students who used the tool to those who did not.

Acknowledgments

The authors thank the Editor-in-Chief and two anonymous referees for constructive comments and suggestions that greatly helped improve the substance and presentation of this paper.

Endnote

[1] Hereafter we call this tool the schedule builder.

References

AIMMS (2015) Gantt chart object. Accessed November 7, 2015, http://main.aimms.com/aimms/user-interface/gantt-chart-object/.

Andresen M, Bräsel H, Engelhardt F, Werner F (2015) LiSA—A Library of Scheduling Algorithms Handbook for Version 3.0. Fakultät für Mathematik Otto-von-Guericke Universität Magdeburg. Accessed November 7, 2015, http://www.math.uni-magdeburg.de/~werner/handbuch-en.pdf.

Baker BM (2005) Computer-aided learning and assessment for spreadsheet modeling of critical path analysis. *INFORMS Trans. Ed.* 6(1):3–12.

Balakrishnan J, Oh SL (2005) An interactive VBA tool for teaching statistical process control (spc) and process management issues. *INFORMS Trans. Ed.* 5(3):19–32.

Barbe WB, Swassing RH, Milone MN (1979) *Teaching Through Modality Strengths: Concepts and Practices* (Zaner-Bloser, Columbus, OH).

Chan TCY (2013) Deal or no deal: A spreadsheet game to introduce decision making under uncertainty. *INFORMS Trans. Ed.* 14(1):53–60.

Enns ST (2008) An interactive spreadsheet-based tool to support teaching design of experiments. *INFORMS Trans. Ed.* 8(2): 55–64.

Kane TJ (2012) *Gathering Feedback for Teaching: Combining High-Quality Observations with Student Surveys and Achievement Gains*. MET Project Policy and Practice Brief. Accessed November 7, 2015, http://metproject.org/downloads/MET_Gathering_Feedback_Practioner_Brief.pdf.

Kydd CT (2012) The effectiveness of using a web-based applet to teach concepts of linear programming: An experiment in active learning. *INFORMS Trans. Ed.* 12(2):78–88.

Leong TY (2007) Simpler spreadsheet simulation of multi-server queues. *INFORMS Trans. Ed.* 7(2):172–177.

Liu Q, Zhang X, Liu Y, Lin L (2013) Spreadsheet inventory simulation and optimization models and their application in a national pharmacy chain. *INFORMS Trans. Ed.* 14(1):13–25.

Manzano RJ (2011) Art and science of teaching/The perils and promises of discovery learning. *Promoting Respectful Schools* 69(1):86–87.

Mason AJ (2013) Solverstudio: A new tool for better optimisation and simulation modelling in excel. *INFORMS Trans. Ed.* 14(1):45–52.

Pinedo ML (2012) *Scheduling: Theory, Algorithms, and Systems*, 4th ed. (Springer, New York).

Pinedo ML, Chao X, Leung J (2002) *LEKIN—Flexible Job-Shop Scheduling System*. Accessed November 7, 2015, http://community.stern.nyu.edu/om/software/lekin/.

Raffensperger JF, Richard P (2005) Implementing dynamic programs in spreadsheets. *INFORMS Trans. Ed.* 5(2):25–46.

Scheubrein RJ, Kulturel-Konak S (2006) An electronic teaching assistant for basic operations management models. *INFORMS Trans. Ed.* 7(1):99–105.

Smartsheet (2015) Online gantt chart software. Accessed November 7, 2015, http://www.smartsheet.com/gantt-chart-software.

Snider B, Balakrishnan J (2013) Lessons learned from implementing web-based simulations to teach operations management concepts. *INFORMS Trans. Ed.* 13(3):152–161.

Sniedovich M (2005) Towards and AOA-free courseware for the critical path method. *INFORMS Trans. Ed.* 5(2):47–63.

Šůcha P, Kutil M, Sojka M, Hanzálek Z (2006) TORSCHE scheduling toolbox for Matlab. *IEEE Comput. Aided Control Systems Design Sympos., CACSD '06 Munich, Germany* (IEEE, Piscataway, NJ), 1181–1186.

TeamWeek (2014) A kinder, gentler gantt chart. Accessed November 7, 2015, https://teamweek.com.

Tsai W, Wardell DG (2006) An interactive excel VBA example for teaching statistics concepts. *INFORMS Trans. Ed.* 7(1):125–135.