Interactive Excel based Gantt Chart Schedule Builder

Abstract

Many scheduling dispatching rules are intuitive and processes which people utilize in every day life. As an 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 have the nice property of being easy to understand, thereby enabling the ability to devise the rule by oneself when given the opportunity to experiment via trial and error. This paper presents an interactive Excel based Gantt Chart Schedule builder which enables students to experiment with building schedules for different single and parallel machine problem instances. Instead of explicitly telling students these common scheduling rules, the schedule builder enables students to discover the rules on their own. Herein, we describe the functionality of the schedule builder, assess its effectiveness via the student perception metric, and provide supplemental teaching materials enabling the use of the schedule builder in a wide variety of classroom environments.

Keywords: Gantt Chart, Scheduling, Excel, Discovery Learning.

1 Introduction

Many of the state of the art scheduling rules and heuristics, often called dispatching rules, are concepts adopted by people 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 received is '*The one due next*'. This decision making procedure is exactly the earliest due date (EDD) rule which is indeed 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 $1||L_{\max}$ (see Pinedo (2012)). The EDD dispatching rule has the nice property that it is optimal for $1||L_{\max}$, but even more so, it is easy to understand and intuitive for a naïve decision maker: you work on the task that has the next earliest due date. There are a portfolio of dispatching rules which have these nice properties and can be conveyed concisely with: next schedule the job which (i) gives you the biggest bang for your buck (i.e., weight for time ratio, see weighted shortest processing time (WSPT)), (ii) has the

smallest processing time (SPT)), (iii) has the longest processing time (LPT), and (iv) proceeds the most amount of future tasks (see largest number of successors (LNS)). While not all concise scheduling dispatching rules are optimal rules, they otherwise have good approximation guarantees.

For an instructor these rules are desirable as a student provided with the appropriate foundational knowledge should be able to experiment and discover these dispatching rules by oneself. If the student doesn't arrive at the rule exactly, he or she should be able to use trial and error to see properties of high performing schedules. To enable this process of trial and error experimentation, we have developed an Excel based interactive Gantt chart Schedule Builder[1]. The purpose of the schedule builder is to compliment traditional lectures in a scheduling course allowing students to build Gantt charts (see Figure 1) and instead of being explicitly told a subset of the common scheduling dispatching rules students are able to discover and come up with the properties on their own.

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

We believe that this experimental process enables students to build confidence in an enhanced discovery teaching environment (see Manzano (2011)). Additionally, the schedule builder compliments traditional lecture, aimed for auditory learners, and enables visual learners to see a specific schedule and kinesthetic learners to touch and move around the jobs on different machines (see Barbe et al. (1979)). Moreover, we have built added functionality to the schedule builder which, when provided access, algorithmically builds a schedule based off of a selected scheduling rule. This functionality can be revealed to students after they have successfully experimented with scheduling problem instances seeking to discover properties and ultimately rules for building high quality schedules.

The schedule builder was first utilized in a 3 credit quarter long course (10 weeks) in Summer 2014 entitled OPER 626/LOGM 631 Scheduling Theory. This course is cross listed in the Operational Sciences and Logistics Management departments. The intended audience is Master's and Doctoral students pursuing degrees in Operations Research and Logistics Management, however enrollment is open to all graduate students and most often students from outside these two departments reside in the Systems Engineering Department. The schedule builder was introduced to the students on the first day of class via a demonstration by the instructor walking through the tool's functionality. The schedule builder specifically focuses on deterministic scheduling problems with a single machine or parallel machine environment.

In the first homework assignment, the students were asked four similar questions. The first part of each question asked the students to utilize the tool to find the optimal schedule for a specific scheduling

[1]We will continue by calling this tool the schedule builder

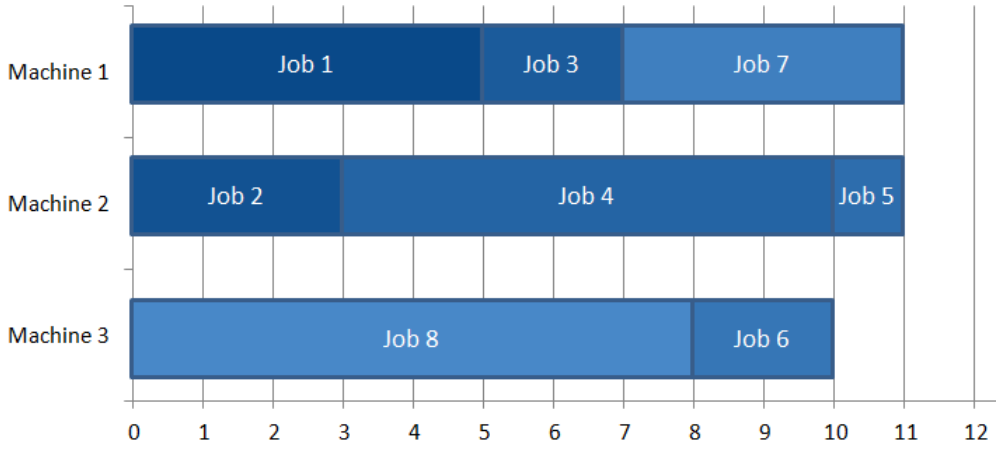

Figure 1: Example of a Gantt chart which the schedule builder enables a user to create.

problem instance. The second part of the each question asked the students to experiment with other instances of the stated problem (i.e., the same machine environment, objective, and constraints but different number of jobs and parameters) and notice properties present across the optimal schedules. The third and final part of each question asked the student 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 to observe the properties of high performing schedules.

Students were asked to provide feedback on the schedule builder after this first homework and throughout the duration of the quarter. Based on the feedback the schedule builder went through thirteen versions. Throughout the rest of the course, the students were not explicitly asked to utilize the tool, however it was used for in class examples and available to students for home and school use due to the tool's development in VBA within Excel.

The effectiveness of the tool was assessed based on the student perception metric (see Kane (2012)). At the end of the course, a questionnaire was given to the students with questions in the following main topic areas: (i) how they utilized the tool and frequency of use, (ii) their perception of the usefulness of the tool, and (iii) ideas for improvement. We present the results and analysis of the answers from this questionnaire in Section 3.

The idea of using electronic tools to aid with teaching is not a new concept. In the context of teaching operations research and management science topics, others have used tools to aid in the instruction of courses with 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 within an Excel environment. A valuable feature of this tool is the Excel environment catering to many users 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 where specific tips on implementing interactive tools in an operations manage-

ment course are discussed by Snider and Balakrishnan (2013). When teaching statistics courses, an instructor is guided towards 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 utilized tools and specific instructional methods for teaching scheduling and project management courses. Baker (2005) created a spreadsheet based tool and Sniedovich (2005) present a specific teaching method 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, and 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 (2014), TeamWeek (2014), and AIMMS (2014)). Further, included with Pinedo (2012), a common book used for teaching a course in scheduling, are three software systems. 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 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 herein. However even if a user were to have access to older operating systems, the book explicitly 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 machine and parallel machine scheduling environments.

A library of Scheduling Algorithms (LiSA) (see Andresen et al. (2014)), also included with Pinedo (2012) is a software package which 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 sequence in time. TORSCHE (see Šůcha et al. (2006)) is another software system eligible for download with the purchase of Pinedo (2012). This system is targeted towards researchers to aid in their development of complex scheduling algorithms. An output of the tool is a Gantt chart. While both of these tools fill a niche within the field of scheduling and visually display a Gantt chart, the purpose of the software packages is not for teaching and does not allow the user to *create* and develop classic scheduling rules on their own through trial and error placement of jobs to machines. Instead these tools are more sophisticated and have built in algorithms which can be used in conjunction with cutting edge new algorithms.

The schedule builder presented herein, embraces the idea of enhanced discovery learning (see Manzano (2011)). With enhanced discovery learning, students are provided with enough knowledge to provide a stepping stone between basic concepts and those which an instructor desires a student to discover. To our knowledge, few others have considered this concept when teaching operations research and management science courses, however Kydd (2012) utilize a interactive applet to teach

the basic and visual aspects of linear programming. The applet enables active learning by allowing both student and teacher 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) development of an Excel-based interactive Gantt chart schedule builder, (ii) explanation of the schedule builder as a learning device for allowing students to *discover* intuitive high quality scheduling rules, (iii) measurement of the effectiveness of the tool through the student perception metric and (iv) dissemination of associated teaching materials which can be utilized in combination with the schedule builder to provide an enhanced discovery environment.

The remainder of this paper proceeds as follows. In Section 2 we provide a detailed description of the schedule builder and its many functions. We assess the effectiveness of the tool and provide suggestions for use within the classroom in Section 3. We conclude in Section 4 with a summary of the main contributions and proposals for future directions.

2 Description of the Tool

In this section, we thoroughly describe the schedule builder tool. This includes 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. The schedule builder is developed in VBA with Excel, therefore the computing requirements for running the tool are Excel 2007, 2010, or 2013 for Microsoft Windows operating systems.

2.1 Input Data

When opening the schedule builder, we first note that the user should enable macros. At this point 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 would not like 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 4 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 divisible

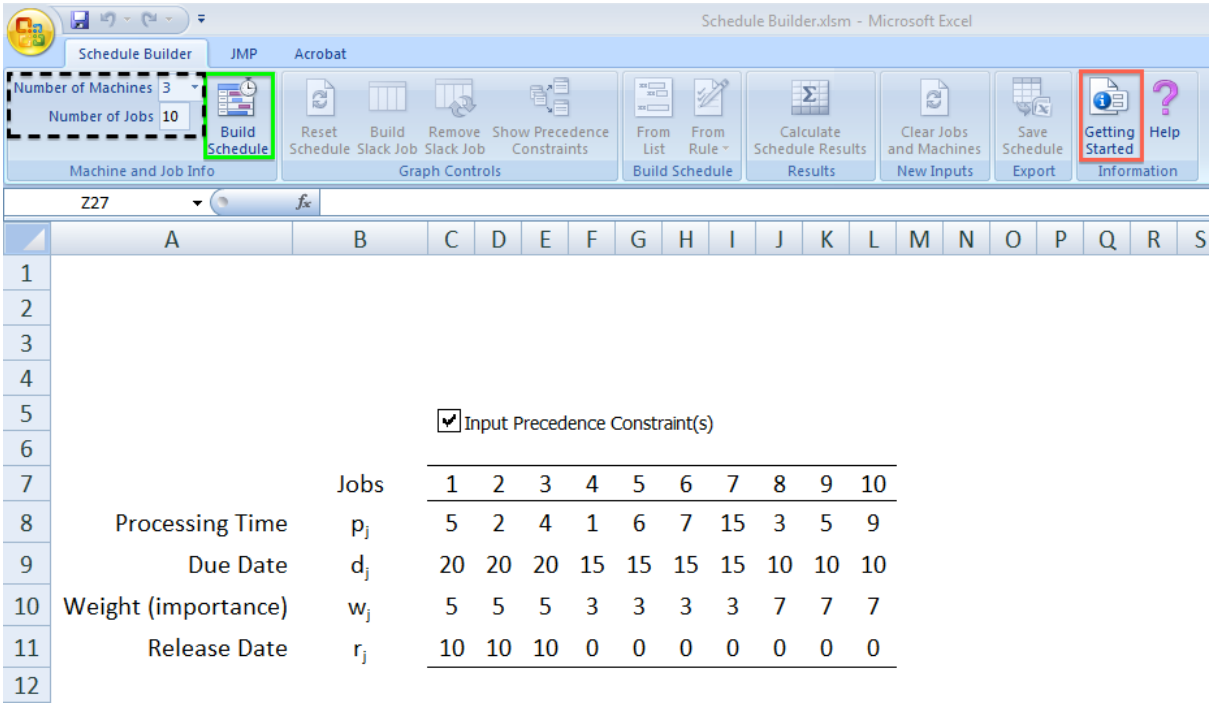

Figure 2: Screenshot of the Main Menu with the input data for Example 1. Further, the *Getting Started* button is outlined in red, Machine and Job number selection in dashed black, and the *Build Schedule* button in green.

by $\frac{1}{2}$ (i.e., 0.5, 1, 1.5, 2, ...). Precedence constraints stipulating that job j must be completed before job k may begin processing are allowed when the precedence check box is selected. Figure 2 shows the input for Example 1 which will be used throughout Section 2. This example considers 3 parallel machines, 10 jobs, precedence constraints, and the parameter values shown in Figure 2. After all data has been input, the user 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 *Build Schedule* button is pressed. 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. In Figure 4, we present a precedence graph for Example 1 where each job is represented as a node and the arrows between the nodes represent the precedence relationships. As an example, from this graph we can conclude that job 5 must be complete before job 4 begins, and also both job 6 and job 9 must be complete before job 7 begins. We have translated this traditional precedence graph into a series of chains in Figure 3.

2.2 Interactive Gantt Chart Functionality

After the *Build Schedule* button is selected and precedence constraints are entered (if applicable), the tool takes you to the interactive Gantt chart maker (see Figure 5). Across the top is the Gantt

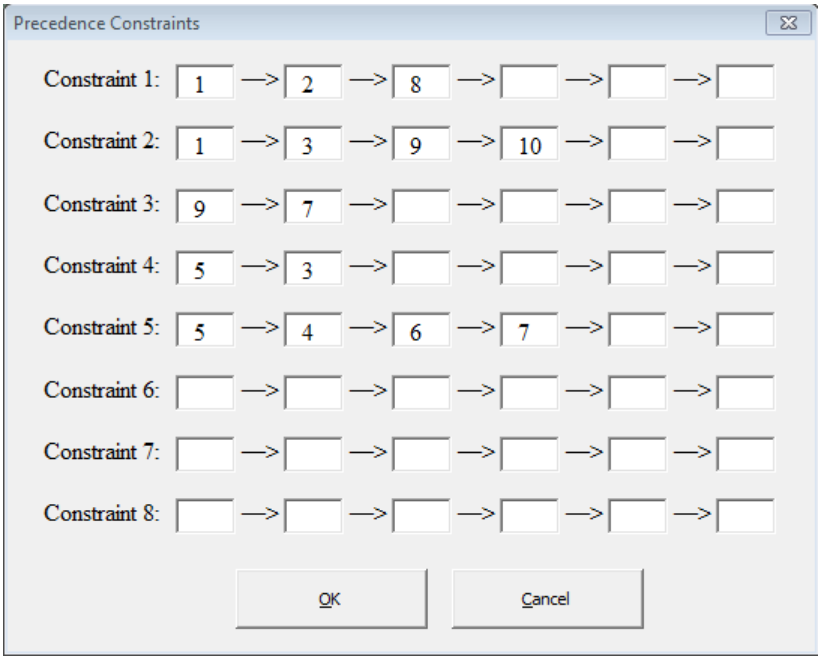

Figure 3: The pop-up which appears after the *Build Schedule* button is pressed if the box indicating precedence is checked. In this pop-up the user can input the precedence relationships and select *OK*, otherwise select *Cancel*.

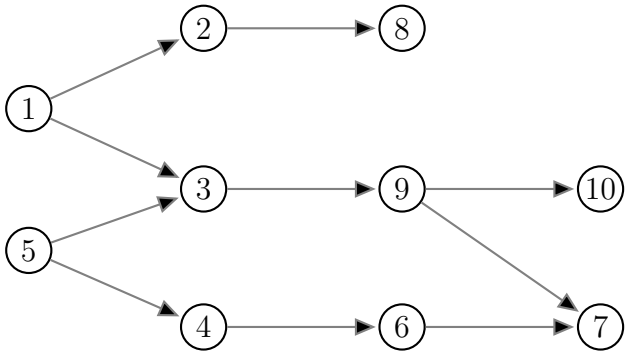

Figure 4: Precedence graph for Example 1 where each job is represented as a node and the precedence is represented by arcs. As an example, both jobs 1 and 5 must be completed prior to starting job 3.

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. Lastly, at the bottom right is a box which displays the objective function value for common scheduling objectives based on the currently built schedule.

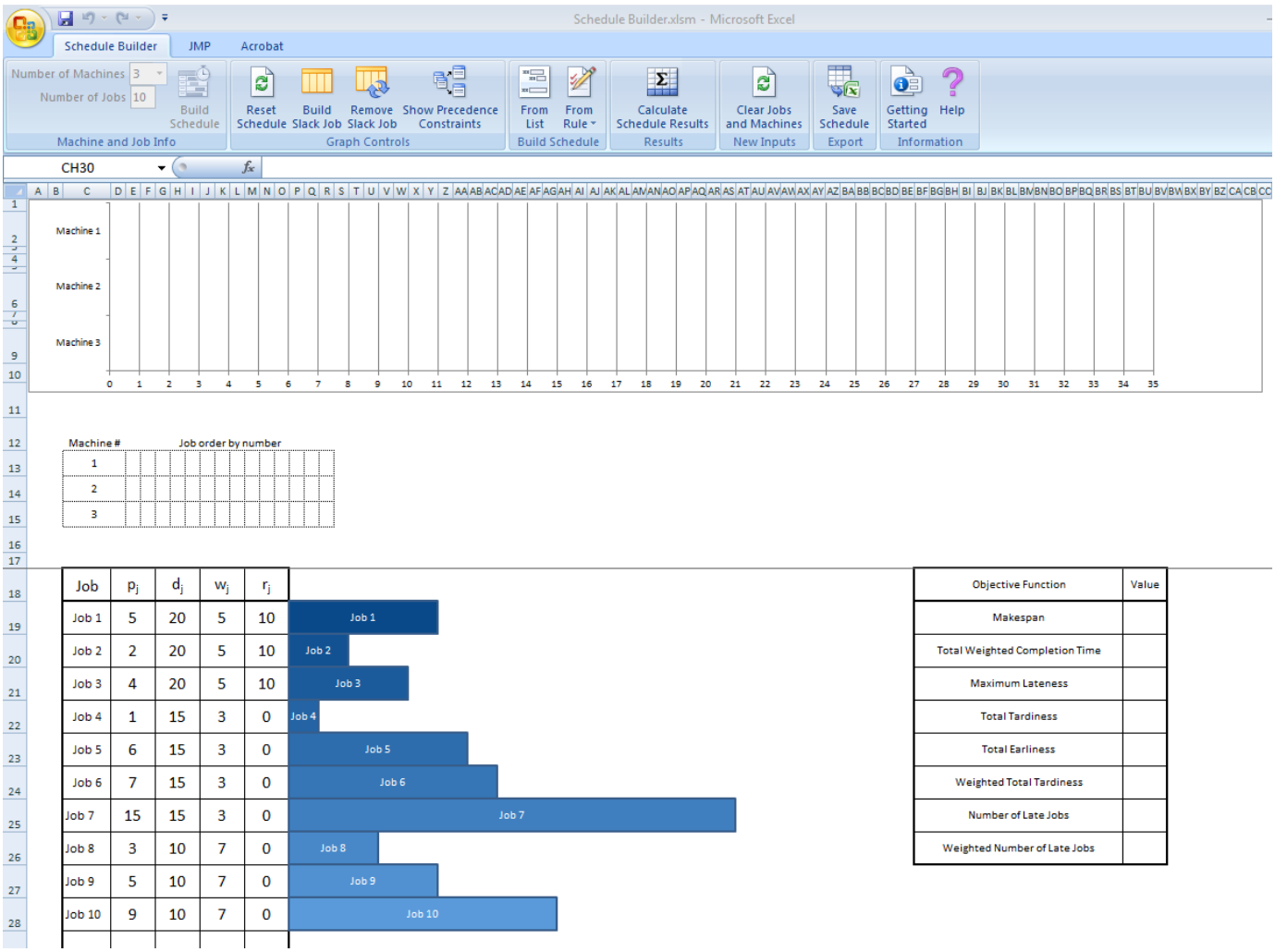

Figure 5: Interactive Gantt Chart Portion of the tool, with the Gantt chart up top, schedule building via a typed list below, jobs with associated parameters at the bottom, and objective function calculations at the bottom right.

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. We note that this *From Rule* functionality can be hidden for the case when students are seeking to discover common scheduling dispatching rules.

The placement of the full rectangles coincides with scheduling jobs in a non-preemptive environment where once a job begins processing it must continue to be processed until it is complete. We have included functionality in the tool 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 sub jobs (e.g., a job j with $p_j = 5$ could have smaller sub jobs 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 2 times 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 0.5 time units. 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. Further, 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 would like to adjust the number of segments or processing times associated with each segment, this series of steps can be repeated. We note, 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 $j.2$, both $j.1$ 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 3 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. We then examine the release dates and 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 your 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. In contrast to a real job, a slack job appears in red for ease of differentiation. We again note, 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 resulting in white space. However, slack jobs are available to explicitly input a delay and are not considered into the objective function calculations. Further, a slack job can be removed by selecting the *Remove Slack Job Button*.

The other functionality 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. The *Calculate Schedule Results* button calculates the set of objectives displayed to the bottom right if all jobs have been placed on the Gantt chart and satisfy the constraints set (i.e., no job is scheduled before its release date and precedence constraints are satisfied). For information regarding the meaning and calculation of these objectives, we point the reader to Pinedo (2012). The *Clear Jobs and Machines* button resets the problem instance, taking the user back to the main menu. Lastly, the *Help* button displays a pop-up box providing detailed descriptions of the schedule builder functionality. When a user is content with a schedule that they have created it can be saved by clicking the *Save Schedule* button where the options to save the schedule to pdf, a new Excel workbook, or an existing Excel workbook are offered. We note, 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 which 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 also the security features of the tool protecting its correct usability. Error checking the feasibility of a built schedule occurs when the user presses the *Calculate Schedule Results* button. At this time a pop-up indicating an error occurs if the schedule has any of the following four characteristics. 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 prior to the completion of job i. A job violates its release date, i.e., if job j is schedule to start at a time $t < r_j$. Two jobs overlap on a machine indicating that a machine is working on more than one job at a time. If at least one job is not scheduled. With this last characteristic, we require all jobs, including all segments of jobs broken up for preemption and all created slack jobs, to be placed on the Gantt chart for the results of a schedule to be calculated. If slack jobs are no longer required the *Remove Slack Job* button should be utilized for elimination.

To ensure the integrity of the tool, we have input security features so that users do not accidentally inhibit the proper calculation and features of the tool. The VBA code which the tool has been built has been password protected. Cells which do not require user input have been locked. This includes 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 seeks to alter the processing time, weight, release date, or due date of any job. Contact information is provided if a user finds bugs or has suggestions for improvement in the *Help* section.

3 Use and Tool Effectiveness

In this section, we discuss ways in which we envision the use of the schedule builder within a course. Further, we present the results of a small assessment capturing the effectiveness of the tool using the student perception metric (see Kane (2012)).

When the schedule builder was first used within the classroom, the students in the course came from a wide variety of backgrounds. There were Master's Operations Research students, Master's Logistics students, Master's Systems Engineering students, and a PhD Operations Research student. We believe the schedule builder helps make differing background less of an issue as students can visualize the concepts being introduced and more readily participate in classroom discussions. While the schedule builder was only utilized in a course with graduate students due the authors' institution, we believe the tool could be easily utilized in an undergraduate course on scheduling or even within a lecture (or lecture sequence) within a deterministic operations research course.

With this paper, we have provided a supplementary assignment which instructors can use to help facilitate an enhanced discovery learning environment for students. The assignment is similar to the one utilized in the Summer 2014 course with modifications made based on observations and feedback. The assignment walks students through 4 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 device a rule for solving instances of this scheduling problem based on the properties they observed in the optimal schedules found during the experimentation phase. The key to an enhanced discovery learning environment is that students are provided with enough foundational knowledge to properly discover concepts. Prior to giving students the assignment provided, they should be exposed to scheduling terminology, as in $\alpha|\beta|\gamma$ notation with possible inputs for each of the triplet components, and introduced to the schedule builder in class where the instructor goes through an example similar to those in the assignment.

Informal favorable feedback was received throughout the quarter, including suggestions for the improvement of the tool. We formally solicited feedback from the students via an anonymous questionnaire in the last week of the course. The questions address how and how frequently students used the tool, their perception of the tool's effectiveness, and suggestions for improvement. The specific

questions asked were:

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 the tool for Homework 1 or did you also use it at other times (e.g., for other homeworks, examples from the book, ...)?
3. Do you believe that 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 the use of the scheduling tool with this course (e.g., use for more in class examples)?

The responses to questions 1 and 2, regarding the frequency and purpose for using the tool, indicated that all students utilized the tool for Homework 1, the homework specifically asking students to devise scheduling rules. The students also indicated that they utilized the tool to verify and visually see schedules they built for other homework assignments and when studying for the final exam. One student did indicate that he preferred using graph paper to build schedules for the course rather than utilizing the tool. Questions 3 and 4 address the perception of students on the usefulness of the tool for understanding concepts. All students in the course indicated that yes the tool was useful, with supplemental feedback saying that it helped to understand how objectives are calculated and the basic concepts in the early stages of the course. Questions 5 and 6 targeted the specific interactive usability of the tool. The students indicated that the usability of the tool increased dramatically from version 1 to version 13 stating that it is intuitive and easy to use. The major comment was the snap to grid feature only occurring after clicking elsewhere in the sheet, which is a feature we are unable to fix within the Excel framework. Further, the students indicated that they would like a feature to save a specific schedule they built to another sheet. We have added this functionality in the final version of the tool. The last question asking for feedback on the use of the tool in the class presented a dichotomy with some students wanting it used for more in class examples, and others liking the use of the tool for independent 'play around'.

4 Conclusions

In this paper, we have described an Excel based interactive Gantt Chart schedule builder which can be used within a course to create an enhanced discovery learning environment. This tool allows students

to use experimentation, through trial and error, to observe properties of high quality schedules for various scheduling rules. This experimentation phase will hopefully enable a student to devise common simple scheduling dispatching rules by themselves instead of being explicitly told. Based on feedback, we believe that this tool is an effective supplement to traditional classroom instruction helping both visual and kinesthetic learners. The tool only considers single machine 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 be considered for inclusion in the tool such as sequence dependent set up times, batch processing, machine speed, and storage on machines. Lastly, while the tool was built in VBA, we have only thoroughly tested the tool on Windows operating systems. Further testing and development should be conducted for building a tool that is compatible with Mac, Linux, and Unix.

References

AIMMS (2014). Gantt chart object. Last accessed on July 21, 2014 at http://www.aimms.com/aimms/user-interface/gantt-chart-object/.

Andresen, M., Bräsel, H., Engelhardt, F., and Werner, F. (2014). *LiSA - A Library of Scheduling Algorithms Handbook for Version 3.0*. Fakultät für Mathematik Otto-von-Guericke Universität Magdeburg. Last accessed on July 21, 2014 at http://www.math.uni-magdeburg.de/~werner/handbuch-en.pdf.

Baker, B. M. (2005). Computer-aided learning and assessment for spreadsheet modeling of critical path analysis. *INFORMS Transactions on Education*, 6(1):3–12. http://dx.doi.org/10.1287/ited.6.1.3.

Balakrishnan, J. and Oh, S. L. (2005). An interactive VBA tool for teaching statistical process control (spc) and process management issues. *INFORMS Transactions on Education*, 5(3):19–32. http://dx.doi.org/10.1287/ited.5.3.19.

Barbe, W. B., Swassing, R. H., and Milone, M. N. (1979). *Teaching through modality strengths: Concepts and practices*. Zaner-Bloser, Columbus, OH.

Chan, T. C. Y. (2013). Deal or no deal: A spreadsheet game to introduce decision making under uncertainty. *INFORMS Transactions on Education*, 14(1):53–60. http://dx.doi.org/10.1287/ited.2013.0104.

Enns, S. T. (2008). An interactive spreadsheet-based tool to support teaching design of experiments. *INFORMS Transactions on Education*, 8(2). http://dx.doi.org/10.1287/ited.1080.0008.

Kane, T. J. (2012). *Gathering Feedback for Teaching: Combining High-Quality Observations with Student Surveys and Achievement Gains*. MET Project Policy and Practice Brief. Last accessed on August 4, 2014 at http://metproject.org/downloads/MET_Gathering_Feedback_Practioner_Brief.pdf.

Kydd, C. T. (2012). The effectiveness of using a web-based applet to teach concepts of linear programming: An experiment in active learning. *INFORMS Transactions on Education*, 12(2):78–88. http://dx.doi.org/10.1287/ited.1110.0076.

Leong, T.-Y. (2007). Simpler spreadsheet simulation of multi-server queues. *INFORMS Transactions on Education*, 7(2):172–177. http://dx.doi.org/10.1287/ited.7.2.172.

Liu, Q., Zhang, X., Liu, Y., and Lin, L. (2013). Spreadsheet inventory simulation and optimization models and their application in a national pharmacy chain. *INFORMS Transactions on Education*, 14(1):13–25. http://dx.doi.org/10.1287/ited.2013.0114.

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

Mason, A. J. (2013). Solverstudio: A new tool for better optimisation and simulation modelling in excel. *INFORMS Transactions on Education*, 14(1):45–52. http://dx.doi.org/10.1287/ited.2013.0112.

Pinedo, M. L. (2012). *Scheduling: Theory, Algorithms, and Systems*. Springer, New York, New York, Fourth edition.

Pinedo, M. L., Chao, X., and Leung, J. (2002). *LEKIN - Flexible Job-Shop Scheduling System*. Last accessed on September 23, 2014 at http://community.stern.nyu.edu/om/software/lekin/.

Raffensperger, J. F. and Richard, P. (2005). Implementing dynamic programs in spreadsheets. *INFORMS Transactions on Education*, 5(2):25–46. http://dx.doi.org/10.1287/ited.5.2.25.

Scheubrein, R. J. and Kulturel-Konak, S. (2006). An electronic teaching assistant for basic operations management models. *INFORMS Transactions on Education*, 7(1):99–105. http://dx.doi.org/10.1287/ited.7.1.99.

Smartsheet (2014). Online gantt chart software. Last accessed on July 20, 2014 at http://www.smartsheet.com/gantt-chart-software.

Snider, B. and Balakrishnan, J. (2013). Lessons learned from implementing web-based simulations to teach operations management concepts. *INFORMS Transactions on Education*, 13(3):152–161. http://dx.doi.org/10.1287/ited.2013.0108.

Sniedovich, M. (2005). Towards and AOA-free courseware for the critical path method. *INFORMS Transactions on Education*, 5(2):47–63. http://dx.doi.org/10.1287/ited.5.2.47.

Šůcha, P., Kutil, M., Sojka, M., and Hanzálek, Z. (2006). TORSCHE Scheduling Toolbox for Matlab. In *IEEE Computer Aided Control Systems Design Symposium (CACSD'06)*, pages 1181–1186, Munich, Germany.

TeamWeek (2014). A kinder, gentler gantt chart. Last accessed on July 20, 2014 at https://teamweek.com.

Tsai, W. and Wardell, D. G. (2006). An interactive excel VBA example for teaching statistics concepts. *INFORMS Transactions on Education*, 7(1):125–135. http://dx.doi.org/10.1287/ited.7.1.125.