A Disk Scheduling Algorithm Simulator

Sukanya Suranauwarat

***Abstract*— This paper presents a simulator that is written in C# and designed as a means of enhancing students' learning of disk scheduling algorithms both in and out of the classroom. The simulator animates the concepts of several disk scheduling algorithms commonly discussed in operating systems textbooks. The simulator has three unique features. First, it uses a more practical model of disk requests that allows new requests to come in while other requests are being processed. Second, it has a practice function that allows the user to reinforce the concepts learnt by solving scheduling problems and check the answers against the simulator. Third, it has a comparison function that lets the user easily compare the simulation results and the performance statistics of different disk scheduling algorithms, up to 9 algorithms, at the same time.**

***Index Terms*—Algorithm Animation, Computer Science Education, Disk Scheduling Algorithms, Educational Software, Operating Systems**

I. Introduction

I have been teaching an operating systems course for over a decade. Based on my teaching experiences, students tend to understand operating system concepts better when explanations are given along with some sort of visualization. In fact, visualization has a long history in computer science education [1-7] and research has shown that carefully designed visualization can beneficially impact students' learning. For example, research [8-12] has shown that active engagement of student attention and students being able to control the pace of their visualization are necessary to make visualization tools educationally effective. Keeping these in mind, I have developed a simulator that animates the concepts of several disk scheduling algorithms commonly discussed in operating systems textbooks.

Besides presenting an animated view of disk scheduling algorithms to the user, the simulator has three unique features. First, it uses a more practical model of disk requests that allows new requests to come in while other requests are being processed. This model is comparatively more practical than the model used in textbook examples, which assumes there are no other disk requests coming while pending requests are being processed. By using a more practical model of disk requests, users will get a better understanding of the fact that the operating system needs to dynamically adjust the disk scheduling queue as a disk request arrives. The user can observe the behavior of disk scheduling algorithms by running them on a set of disk requests, which can be configured easily using the user-friendly interface of the simulator. The second unique feature of the simulator lies in its practice function that allows the user to make his or her own scheduling decisions and check the answers against the simulator. The disk scheduling algorithms are not hard to understand, but they can easily confuse students because they are so similar. Using the practice feature of the simulator will not only help students reinforce the concepts studied but also discern the differences and the similarities among various algorithms. The third unique feature of the simulator enables the user to conveniently compare the simulation results and the performance statistics of different disk scheduling algorithms, up to 9 algorithms, at the same time. The simulation results, as well as the performance statistics of each algorithm, can be saved as an image, which can be viewed or printed by any standard image viewer for further study. The simulator can be used as a means of enhancing students' learning of disk scheduling algorithms both in and out of the classroom.

The remainder of this paper is organized as follows. In section 2, I discuss related work. In section 3, I give a brief overview of the disk scheduling algorithms supported by the simulator. In section 4, I describe the operation of the simulator in detail. In section 5, I describe the uses of the simulator both in and out of the classroom and summarize this work in section 6.

II. Related Work

In this section, some animation tools for learning disk scheduling algorithms that others have developed are discussed.

English and Rainwater [13] conducted research on the effectiveness of using animations as primary instruction tools for teaching an operating systems course. As part of this research project, several animations are developed using Adobe Flash including the animations for teaching FCFS, SSTF, and SCAN disk scheduling algorithms [14]. All of these animations are built upon the examples in

Manuscript received November 9, 2016.
Manuscript revised October 19, 2017.
Sukanya Suranauwarat is with the Computer Science Department, Graduate School of Applied Statistics, National Institute of Development Administration (NIDA) Bangkok, Thailand (e-mail: sukanya@as.nida.ac.th).

their adopted textbook, and the user is not allowed to create his or her own set of disk requests or specify his or her own configuration parameters. As a result, the user can only observe the behavior of disk scheduling algorithms in the same scenario as in the adopted textbook. These animations demonstrate how an algorithm works in a similar manner as the simulator does. The total disk head movement or the total seek length is also calculated.

Track Animation [15] is a Windows application that simulates the following disk scheduling algorithms: FCFS, SSTF, SCAN, C-SCAN, LOOK, and C-LOOK. It is available under Creative Commons Attribution License. It is similar to those developed by English and Rainwater in terms of functionality. But it is more flexible in the sense that it allows the user to create his or her own set of disk requests or use a randomly generated set of disk requests.

Meyer and Verdicchio [16] developed several programs in the form of executable Java JAR files to animate various concepts presented in an introductory computer science course including disk scheduling. Their disk scheduling program supports three scheduling algorithms: FCFS, SSTF, and LOOK. The user can type in a set of disk requests in a text box of the program or use the predefined one. The program presents a disk with 100 cylinders as a tall yellow bar where cylinder 0 is at the top of the yellow bar and cylinder 99 is at the bottom of the yellow bar, and presents the disk head as a thin red bar that moves up and down inside the yellow bar. Exactly where the red bar moves to next is the subject of the scheduling algorithm. Using this representation, it becomes very hard to track the order in which cylinders are serviced. When the simulation is over, nothing is reported to the user to give any valuable insight about the algorithms.

None of the tools above have a functionality to compare the performance statistics of different disk scheduling algorithms or a capability to use a more practical model of disk requests that allows new requests to come in while other requests are being processed. To my knowledge, the only animation tool that allows new disk requests to come in while other requests are being processed is the one developed by Robbins [17]. This tool shows the animation of the disk head movement for various algorithms including FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK, and FSCAN. The tool represents a scan of the disk in one direction as a horizontal line, with tick marks at the positions of accessed cylinders. The user can get a general idea of how the disk head moves for each algorithm but cannot tell exactly the order in which the cylinders are serviced because the tool does not display any cylinder number at all. The reason behind this could be that it is impossible to display all the accessed cylinder numbers in a readable manner when a large set of requests is used. Although this tool can be used to show the animation of the disk head movement for each algorithm, its main usage is for performance comparison and analysis. Therefore, it is not unusual to run this tool using a large set of disk requests. In this sense, this tool is not a good candidate for teaching disk scheduling concepts through animation. To use a more realistic set of disk requests with this tool, a first arrival time, a distribution of interarrival times, and distribution of cylinder references needs to be specified and saved into a file. While the model of disk requests adopted by the simulator is not as realistic as Robbins' tool, it is a simpler model that is easier to understand and configure, and still satisfies the goal of teaching disk scheduling algorithms through animation.

Lastly, none of the tools above provide similar functionality as the practice mode of the simulator that allows the user to do practice tests, and as the comparison mode of the simulator that allows the user to simulate up to 9 algorithms at the same time.

III. Overview of Disk Scheduling Algorithms

When several processes are trying to access data from the disk concurrently, the operating system will use a disk scheduling algorithm to determine which disk request should be serviced next. Textbooks usually discuss only the traditional disk scheduling algorithms that concentrate on reducing seek times for a set of disk requests. The seek time is the time it takes the disk head to move from the current position to the cylinder containing the desired sector. Typical algorithms that are discussed in textbooks [18-23] and supported by the simulator are listed below.

- **FCFS (First-Come-First-Served)** algorithm services the disk requests in the order in which they arrive. Therefore, a request's position in the queue is unaffected by arriving requests. This ensures that a request cannot be postponed indefinitely, but it also means that FCFS might perform a long-winded seek operation to service the next request, even though another request in the queue is closer and can be serviced faster.
- **SSTF (Shortest-Seek-Time-First)** algorithm chooses the next request with the least seek time from the current head position. This algorithm can lead to indefinite postponement because its seek pattern tends to be highly localized, which can lead to poor response times for requests to the innermost and outermost tracks.
- **SCAN** algorithm moves the disk head from one end of the disk to the other, servicing requests along the way. When the disk head reaches the other end of the disk, it is moved in the reverse direction and newly arrived requests are processed in a reverse scan. In this sense, it is called the elevator algorithm because an elevator continues in one direction servicing requests before reversing direction.

- **C-SCAN (Circular SCAN)** algorithm is a variant of SCAN that is designed to provide a more uniform wait time. Like SCAN, C-SCAN moves the disk head from one end of the disk to the other, servicing requests along the way. However, it never performs a reverse scan. When the disk head reaches the other end, C-SCAN moves the disk head back to that end of the disk from where it started (without servicing requests in between) and initiates another scan.
- **LOOK** algorithm is a variant of SCAN that "looks" ahead to see if there are any requests pending in the direction of the disk head movement. If there are no such requests, then the disk head will be reversed to the opposite direction and requests on the other direction can be served.
- **C-LOOK (Circular LOOK)** algorithm is a variant of LOOK algorithm that behaves in the same manner as C-SCAN, except it moves the disk head only as far as needed to service the last request in a scan before starting another scan.
- **FSCAN** algorithm is a variant of SCAN that freezes the queue to be serviced when it is doing a scan of the disk and places requests that arrive during the scan into a queue to be serviced later. This algorithm tries to avoid starvation of far-away requests by delaying the service of late-arriving but nearer by requests.
- **Pickup** algorithm is a variant of FCFS. In this algorithm, the requests are generally taken in order as with FCFS, but as the system is moving the disk head it will stop for any tracks that are being passed over that have a request in the queue.
- **LIFO** (Last-In-First-Out) algorithm always chooses to service the last request first.

IV. The Simulator

The disk scheduling algorithm simulator is written in C# and designed to be intuitive, engaging; and easy to use and control. It has three operating modes which are simulation, practice, and comparison modes. Each mode is described below.

A. Simulation Mode

While in this mode, the user can watch the animation showing how a disk scheduling algorithm works. The user can select an algorithm of his or her interest through a drop-down menu at the top-left section of the simulator, as shown in Fig. 1. Figs. 1 and 2 are screenshots of the simulator in simulation mode. Next to the drop-down menu is the "Concept" button. When this button is clicked, a window containing a description along with a scheduling example of the currently selected algorithm will pop up. By clicking the "Draw" button, the user can start watching the animation of the selected algorithm in the bottom half of the simulator immediately. In this case, the default set of disk requests and the default configuration parameters will be used. The top-right section of the simulator shows the values of the configuration parameters that are currently set and used. The configuration parameters include "Initial Head Position", "Direction", "Milliseconds/Cylinder", "Min Cylinder", and "Max Cylinder". The "Initial Head Position" parameter is used to specify the position of the disk head when the scheduling starts. For SCAN algorithm and its variants, the direction of the disk head movement also needs to be specified, which can be done through the "Direction" parameter. The value of "<< L" of the "Direction" parameter indicates that the disk head is initially moving toward the lowest-numbered cylinder (i.e., the outermost cylinder at the edge of the disk), whereas the value of "R >>" indicates that the disk head is initially moving toward the highest-numbered cylinder (i.e., the innermost cylinder nearest the spindle). The lowest-numbered and the highest-numbered cylinders are specified through the "Min Cylinder" and the "Max Cylinder" parameters respectively. The "Milliseconds/Cylinder" is used to specify the time to move the disk head over a cylinder; it will be used to calculate the seek time which will be described later.

Textbook examples typically assume that there is a set of pending disk requests, and while these requests are being processed, no other requests come in. To use the simulator with this model of disk requests, the user can simply specify a set of cylinder numbers to be accessed into the "Disk Requests" text box. The user can separate each cylinder number by a comma or space. For example, "1, 36, 16, 34, 9, 12" and "1 36 16 34 9 12" are both valid inputs to the simulator. Alternatively, the user can use a randomly generated set of disk requests by clicking on the "Random" button, and then specifying the total number of disk requests the user wants the simulator to generate. The randomly generated cylinders will be in the ranges specified by the "Min Cylinder" and the "Max Cylinder" parameters. The user can save any specified set of disk requests and configuration parameters in a file by clicking the "Save" button. Later on, the user can reuse any saved set of disk requests and configuration parameters by clicking the "Open" button followed by the name of the previously saved file. The user has an option to save several sets of disk requests and configuration parameters in the same file or in a different file. To give the user opportunities to explore the disk scheduling algorithms in a more practical situation, the simulator allows the user to specify the disk requests that arrive while other requests are being processed. To use the simulator with this model of disk requests, the user needs to specify the cylinders to be accessed in the form of x/y where x is the disk request for cylinder numbered x that arrives while the cylinder numbered y is being processed.

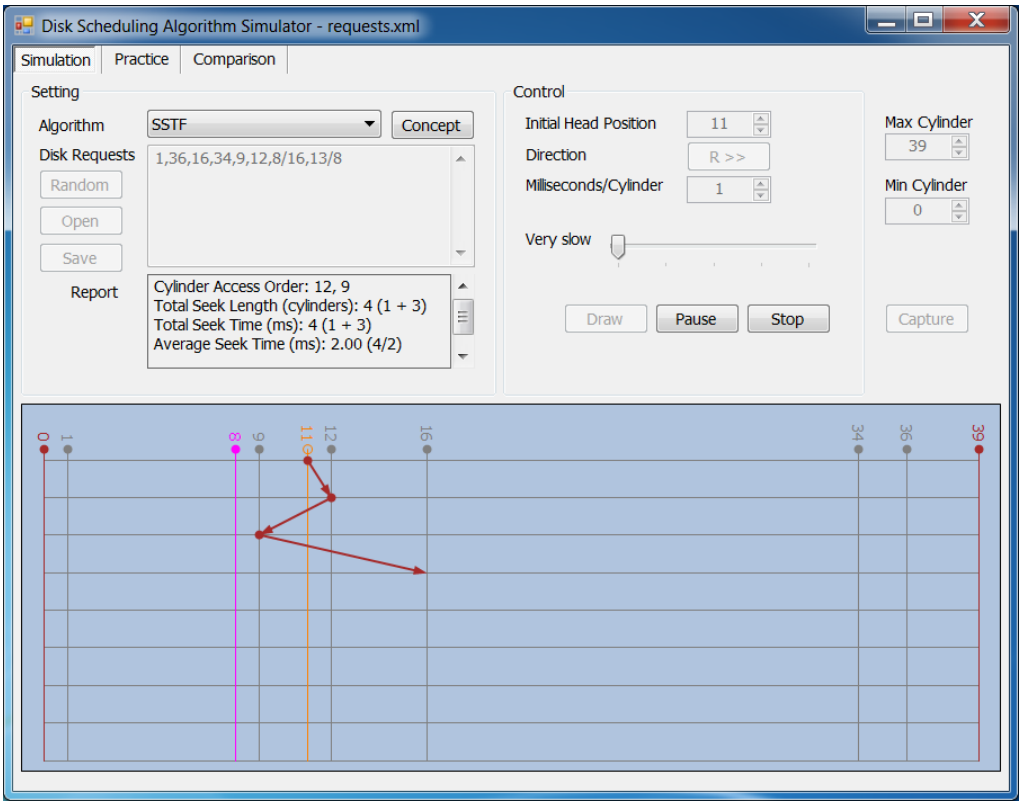


Fig. 1. A screenshot of simulation mode when the request for cylinder 8 arrives while the request for cylinder 16 is being processed.

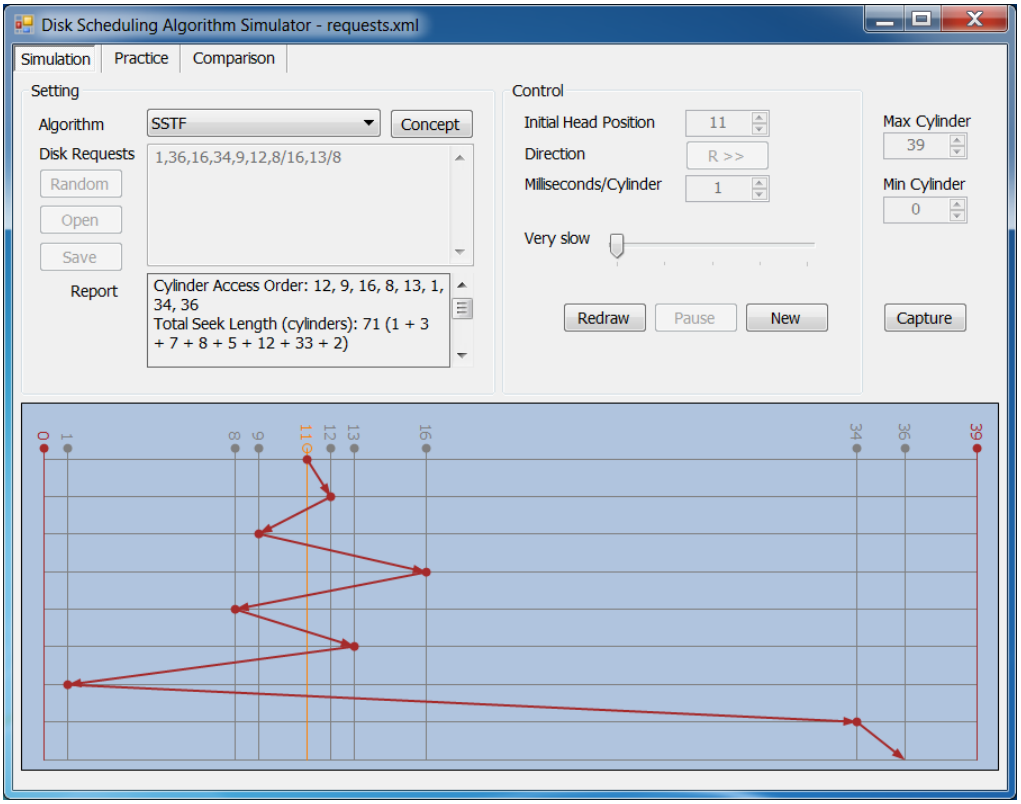


Fig. 2. A screenshot of simulation mode when the simulation is over.

In the simulation of Fig. 1, the SSTF algorithm was selected and a user-defined set of disk requests was used. The user-defined set of disk requests is “1, 36, 16, 34, 9, 12, 8/16, 13/8”. When the simulation starts, there is a queue of pending disk requests for the following cylinders: 1, 36, 16, 34, 9 and 12, and the disk head is at cylinder 11. The requests for cylinders 8 and 13 arrive while the requests for cylinders 16 and 8 are being processed respectively. The simulator shows the disk requests as they arrive and lets the user know about their presence by temporarily displaying them in an outstanding color of pink and making them blink a couple of times. As shown in Fig. 1, the request for cylinder 8 arrives while the request for cylinder 16 is being processed; at that time, the request for cylinder 13 has not yet entered the queue. When the current request is finished, the operating system examines the requests and decides which request to handle next. Using the SSTF algorithm, it will handle the closest request next. The simulator will show the animation of the disk head moving seamlessly from the current request to the next one. If the speed of the animation is not right, the user can adjust it using the speed-control slider, which is located under the configuration parameters. Under the speed-control slider is a set of buttons for controlling the animation.

In the simulation of Figs. 1 and 2, if no other requests arrive while the requests in the pending queue are being processed, the order in which the cylinders are serviced would be 12, 9, 16, 1, 34, and 36. In fact, while the request for cylinder 16 is being processed, a new request for cylinder 8 is present, that request will have priority over cylinder 1. The request for cylinder 13 then comes in, causing the disk head to go to cylinder 13 next instead of cylinder 1. Seeing such an example that allows more requests to come in while other requests are being processed will help students, in this case, foresee the problem when using SSTF with a heavily loaded disk. That is, the disk head will tend to stay in the middle of the disk most of the time; hence requests at either extreme will have to wait until a statistical fluctuation in the load causes there to be no requests near the middle [18]. Requests far from the middle may get poor service.

When the simulation is over as shown in Fig. 2, the order in which the cylinders are serviced is 12, 9, 16, 8, 13, 1, 34, and 36, as shown in the “Report” text box. With this order, the disk head movements are 1, 3, 7, 8, 5, 12, 33, and 2, for a total disk head movement of 71 cylinders. The total disk head movement or the total seek length, along with the total seek time and the average seek time are also shown in the “Report” text box. These are common criteria for evaluating the performance of the traditional algorithms aimed to reduce seek times. The simulator also shows how to calculate these values in the parentheses next to them. Operating systems textbooks often assume that the time to seek between two cylinders was proportional to the number of cylinders moved. This is a simple model that suffices for the purposes of analyzing traditional algorithms, but it does not accurately model seeking on real disks. The simulator will calculate the total seek time of each algorithm using this model, which results in the product of the total seek length and the value of “Milliseconds/Cylinder” parameter. By clicking the “Capture” button, the simulation result in graphic and the performance statistics can be saved as an image that can be viewed or printed by any standard image viewer program for further study.

B. Practice Mode

While in this mode, the user can reinforce the concepts learnt by solving scheduling problems and check the answers against the simulator. Figs. 3 and 4 are screenshots of the simulator in practice mode. Following usability best practices, the look and feel of practice mode has been designed to be as similar to simulation mode as possible. As in simulation mode, the user needs to select an algorithm of his or her interest, and specify a set of disk requests and configuration parameters or use the default ones.

In Fig. 3, the LOOK algorithm was selected, the initial head position is 53, the disk head is moving toward cylinder 199, and the queue of pending disk requests contains 98, 183, 37, 122, 14, 124, 65, and 67. The major difference between simulation and practice modes is that the bottom half of simulator in practice mode does not display the animation of how the selected algorithm works but is the interface for the user to enter his or her scheduling decisions. To be more specific, the user can decide which request will be serviced next by clicking on the cylinder number the user thinks should be accessed next. The cylinder number between the lowest-numbered and the highest-numbered cylinders will appear when the user moves the mouse pointer over it as shown in Fig. 3. After the user selects the cylinder which will be accessed next by clicking on it, the simulator will move the disk head from the current cylinder (67) to the selected one (which in this case is 98). The user can undo his or her scheduling decision by simply dragging the selected cylinder number to anywhere outside the practice area. The user can also change his or her scheduling decisions by simply clicking on the previously selected cylinder number and moving it to the location of the cylinder number the user thinks it should be. When finished, the user can click the “Answer” button to check if the answer is right. As shown in Fig. 4, the user is confused about the direction of the disk head after it services all the requests in the initial direction. This is a typical mistake my students make.

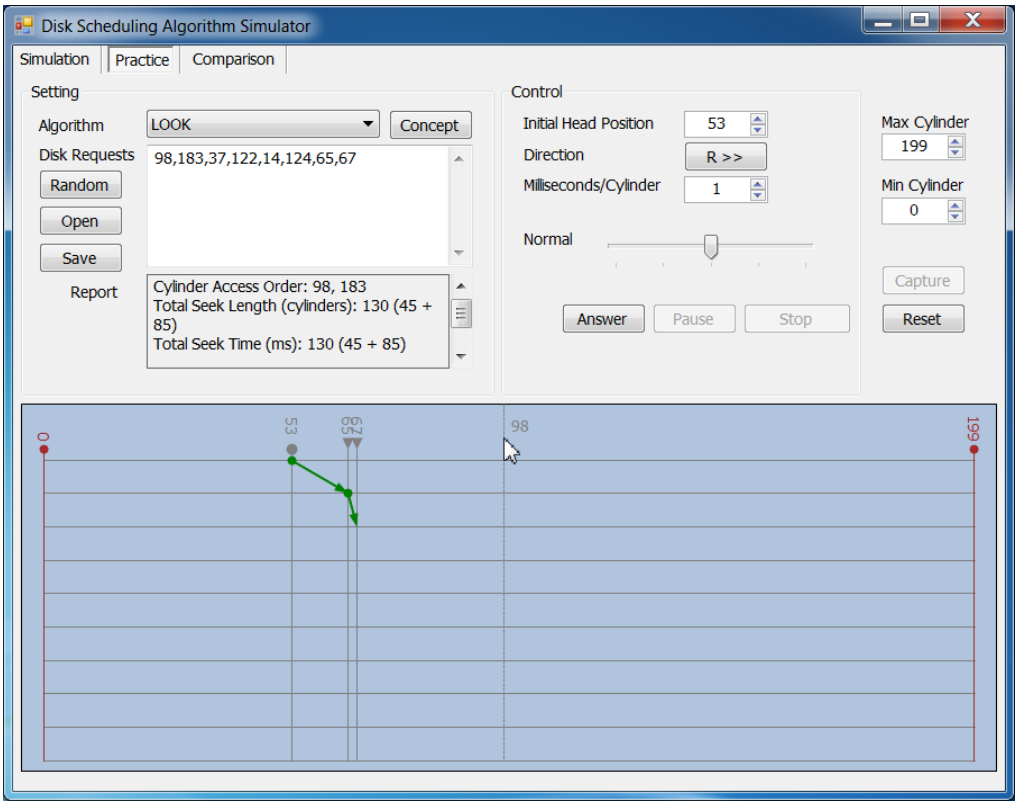


Fig. 3. A screenshot of practice mode when LOOK algorithm is selected and six requests are pending.

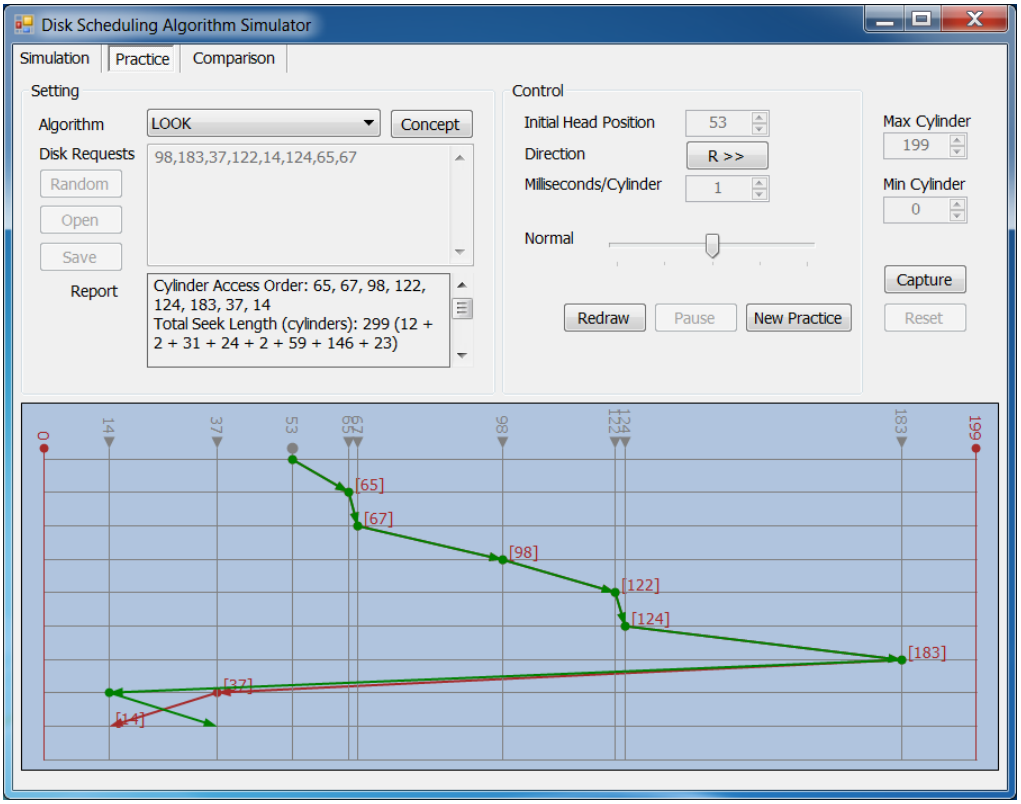


Fig. 4. A screenshot of practice mode after the “Answer” button is clicked.

C. Comparison Mode

The disk scheduling algorithms can be evaluated by running them on a particular set of disk requests and computing the total seek length, the total seek time, and the average seek time. In comparison mode, the user can compare the performance of all the algorithms at the same time, as shown in Fig. 5.

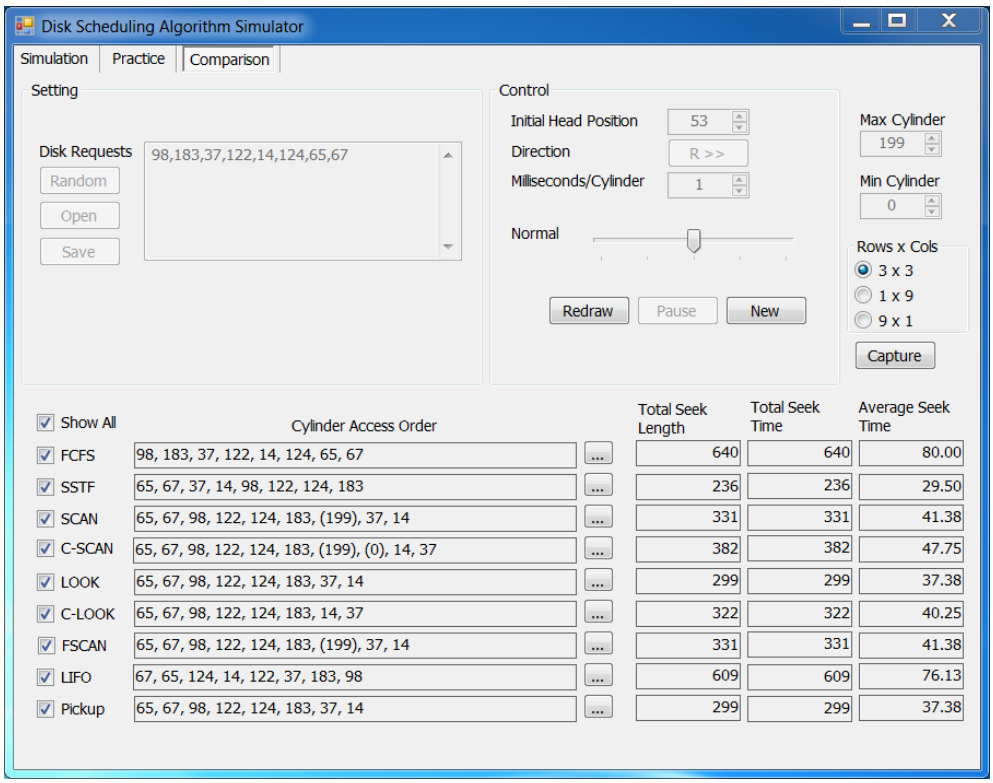


Fig. 5. A screenshot of the performance statistics of all the algorithms in comparison mode.

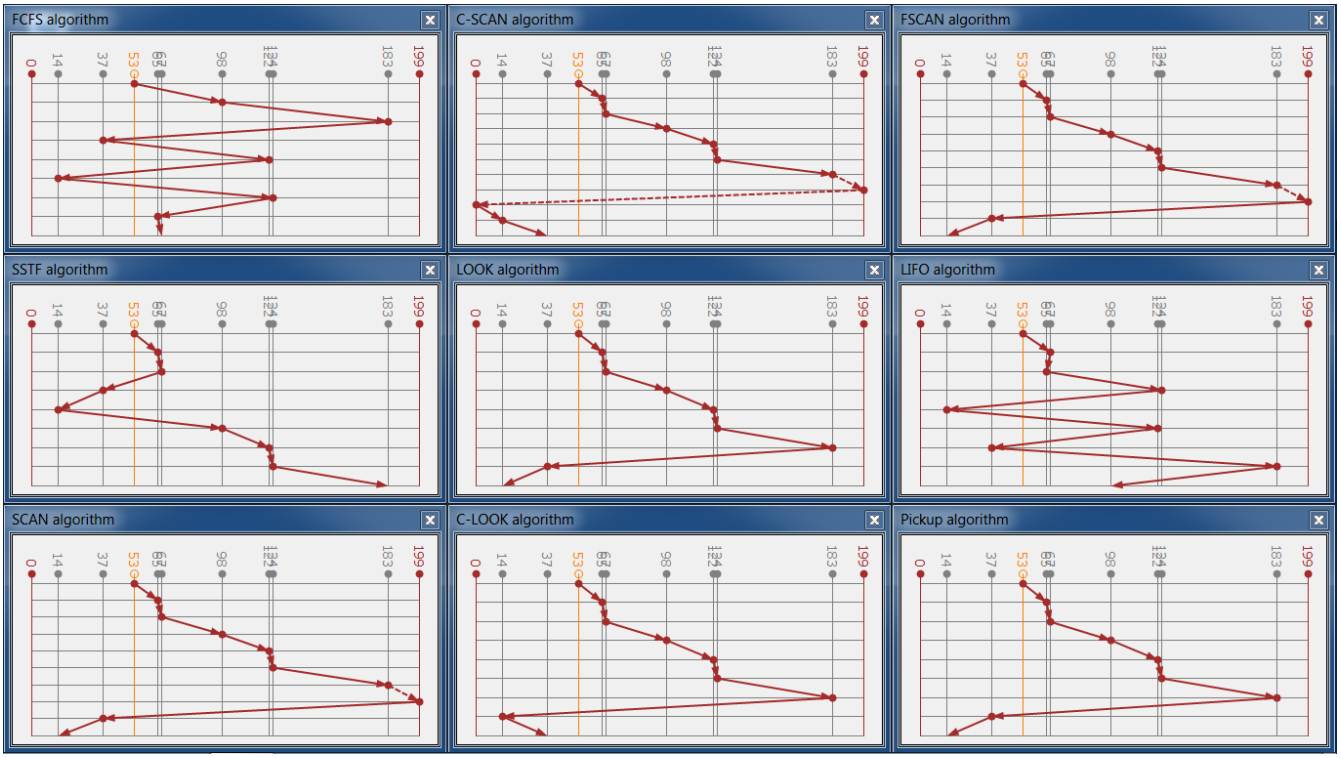


Fig. 6. A screenshot of the simulation results of all the algorithms in comparison mode.

As in the simulation and the practice modes, the user can use the default set of disk requests, a user-defined set of disk requests, or a randomly generated set of disk requests. In Fig. 5, there is a queue of pending disk requests for the following cylinders: 98, 183, 37, 122, 14, 124, 65, and 67, the initial head position is 53, and the disk head is moving to toward cylinder 199. By clicking on the "Show All" checkbox in Fig. 5, the user can also watch the simulations of all the algorithms at the same time, as shown in Fig. 6. Alternatively, the user can select only the algorithms of his or her interest by clicking the checkboxes in front of them in Fig. 5. As in the simulation mode, the simulation results and the performance statistics can be saved as an image by clicking on the "Capture" button with a couple options of how the algorithms will be arranged in the saved image in terms of rows and columns, as shown in Fig. 5.

As described in section 2, SCAN, C-SCAN, and FSCAN algorithms move the disk head across the full width of the disk, even though there may not be any request for the cylinder at either end of the disk. When there are no requests for cylinders at either end of the disk, the simulator will show the disk head movement toward either end in a dotted line to make sure that students understand the reason why the disk head is moving over there, as shown in Fig. 6. For the same reason, when the simulator reports the order in which the cylinders are serviced using these algorithms, although the cylinder at either end of the disk will be included, it will be differentiated from the real disk requests by being put in parenthesis, as shown in Fig. 5.

V. Use of the Simulator

The simulator can be used both in and out of the classroom. In the classroom, instructors can use the simulator to demonstrate traditional disk scheduling algorithms. Instructors can also demonstrate the situation where new disk requests come in while other requests are being processed, which is beyond normal textbook examples. On the other hand, the students can work, experiment, and do more scheduling problems with the simulator out of the classroom. The simulator can also be used for self-study of traditional disk scheduling algorithms. This is useful when class time is tight or when modern disk technology that is not covered in the textbook should be discussed in the classroom.

VI. Conclusion

This paper presents an intuitive, engaging, and easy-to-use simulator that animates the concepts of traditional disk scheduling algorithms. The simulator has the following operating modes: simulation, practice, and comparison. The simulation mode animates how a disk scheduling algorithm works. The practice mode allows the user to make his or her own scheduling decisions by specifying the order each disk request will be served and then checking if the answers are right. The comparison mode allows the user to compare the performance of different algorithms in terms of the total seek length, the total seek time, and the average seek time. The user can also simulate up to 9 algorithms at the same time. The simulator supports both simple and practical models of disk requests. In the classroom, instructors can use the simulator to demonstrate traditional disk scheduling algorithms using either or both models of disk requests. Outside the classroom, students can work, experiment, and do more scheduling problems with the simulator. In the future, the simulator's impact on student learning will be assessed.

References

[1] J. Cross, D. Hendrix, L. Barowski, and D. Umphress, "Dynamic Program Visualizations: an Experience Report," in Proceedings of the 45th ACM Technical Symposium on Computer Science Education, 2014, pp. 609-614.

[2] V. Karavirta and C. A. Shaffer, "JSAV: the JavaScript Algorithm Visualization Library," in Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education, 2013, pp. 159-164.

[3] V. Manickam and A. Aravind, "If a Picture is Worth a Thousand Words, What would an Animation be Worth?," in Proceedings of the 16th Western Canadian Conference on Computing Education, 2011, pp. 28-32.

[4] C. A. Shaffer, M. L. Cooper, A. J. D. Alon, M. Akbar, M. Stewart, S. Ponce, and S. H. Edwards, "Algorithm Visualization: the State of the Field," ACM Transactions on Computing Education, vol. 10, no. 3, Article 9, 2010.

[5] C. Smith, J. Strauss, and P. Maher, "Data Structure Visualization: The Design and Implementation of an Animation Tool," in Proceedings of the 48th Annual Southeast Regional Conference, 2010, Article No. 72.

[6] J. Edgar and T. Donaldson, "A novel Sorting Animation: Permuting Picture Pixels," in Proceedings of the 14th Western Canadian Conference on Computing Education, 2009, pp. 29-33.

[7] M. Krebs, T. Lauer, T. Ottmann, and S. Trahasch, "Student-built Algorithm Visualizations for Assessment: Flexible Generation, Feedback and Grading," in Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, 2005, pp. 281-285.

[8] J. Urquiza-Fuentes and J. Ángel Velázquez-Iturbide, "A Survey of Successful Evaluations of Program Visualization and Algorithm Animation Systems," ACM Transactions on Computing Education - Special Issue on the 5th Program Visualization Workshop, vol. 9, no. 2, Article No. 9, 2009.

[9] P. Saraiya, C. Shaffer, D. Mccrickard, and C. North, "Effective Features of Algorithm Visualizations," in Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, 2004, pp. 382-386.

[10] S. Grissom, M. McNally, and T. Naps, "Algorithm Visualization in CS Education: Comparing Levels of Student Engagement," in Proceedings of the 2003 ACM Symposium on Software Visualization, 2003, pp. 87-94.

[11] C. Hundhausen, S. Douglas, and J. Stasko, "A Meta-Study of Algorithm Visualization Effectiveness," Journal of Visual Languages and Computing, vol. 13, no. 3, pp. 259-290, 2002.

[12] M. Byrne, R. Catrambone, and J. Stasko, "Evaluating Animations as Student Aids in Learning Computer Algorithms," Computers & Education, vol. 33, no. 4, pp. 253-278, 1999.

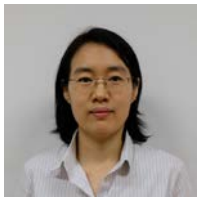
[13] B. M. English and S. B. Rainwater, "The Effectiveness of Animations in an Undergraduate Operating Systems Course", Journal of Computing Sciences in Colleges, Vol. 26, No. 5, pp. 53-59, 2006.

[14] B. M. English and S. B. Rainwater, *COSC 3355 Animations.* [Online]. Available: http://cs.uttyler.edu/Faculty/Rainwater/COSC3355/Animations/index.htm. Accessed on: October 5, 2016.

[15] J. Zandueta, *Track Animation*. [Online]. Available: https://sourceforge.net/projects/trackanimation/. Accessed on: October 5, 2016.

[16] R. M. Meyer and M. Verdicchio, "Disk Scheduling," in *Explorations in Computer Science*, 3rd ed., Jones & Bartlett Learning, 2016, pp. 146-148.

[17] S. Robbins, “A Disk Head Scheduling Simulator,” in Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, 2004, pp. 325-329.
[18] A. Tanenbaum, “Disk Arm Scheduling Algorithms,” in *Modern Operating Systems*, 4th ed., Prentice Hall, 2014, pp. 379-382.
[19] W. Stallings, “Disk Scheduling,” in *Operating Systems: Internals and Design Principles*, 7th ed., Prentice Hall, 2012, pp. 487-494.
[20] A. Silberschatz, P. Galvin, and G. Gagne, “Disk Scheduling,” in *Operating System Concepts*, 9th ed., John Wiley & Sons, 2012, pp. 472-478.
[21] J. M. Garrido, R. Schlesinger, K. Hoganson, “Hard Disk I/O Scheduling,” in *Principles of Modern Operating Systems*, 2nd ed., Jones & Bartlett Learning, 2011, pp. 248-251.
[22] G. Nutt, “Optimizing Access on Magnetic Disks,” in *Operating Systems*, 3rd ed., Addison Wesley, 2004, pp. 182-183.
[23] H. M. Deitel, P. J. Deitel, D. R. Choffnes, “Disk Scheduling Strategies,” in *Operating Systems*, 3rd ed., Prentice Hall, 2004, pp. 533-542.



Sukanya Suranauwarat received her B.Eng., M.Eng., and Ph.D. in Computer Science from Kyushu University, Japan in 1997, 1999, and 2002 respectively. Since then, she has been teaching at Graduate School of Applied Statistics, National Institute of Development Administration (NIDA), Thailand. Her research interests include computer science education, operating systems, and web technologies.