Extending The Effectiveness of Algorithm Visualization with Performance Comparison through Evaluation-integrated Development

by Felix Christian Jonathan Oscar Karnalim, Mewati Ayub

Submission date: 12-Aug-2021 12:34PM (UTC+0700)
Submission ID: 1630516550
File name: 005_Extending_The_Effectiveness_of_Algorithm.pdf (1.36M)
Word count: 4531
Character count: 26533

Extending The Effectiveness of Algorithm Visualization with Performance Comparison through Evaluation-integrated Development

Felix Christian Jonathan[#1], Oscar Karnalim[#2], Mewati Ayub[#3]
Information Technology, Maranatha Christian University
Bandung, Indonesia
[1]felix.c.jonathan@outlook.com
[2]oscar.karnalim@gmail.com
[3]mewati@itmaranatha.org

***Abstract*—Since several undergraduate CS students cannot understand Algorithm topic clearly due to algorithm complexity and limited class duration, several Algorithm Visualization (AV) for teaching algorithms have been developed. However, since most AV only focus on visualizing algorithm steps without mentioning why that algorithm should be chosen based on given problem, students cannot improve their understanding further than Application level (based on Bloom taxonomy). In this paper, we extend the capabilities of AV by utilizing case-based performance comparison. Case-based performance comparison aim to let students differentiate several algorithm and improve their understanding further. Additionally, we utilize evaluation-integrated development since the main goal of an AV is not only technical functionality but also its usability. For our implementation, we implement these aspects to algorithm for solving classic problems such as 0/1 knapsack and Minimum Spanning Tree (MST) problem.**

Keywords—algorithm visualization; performance comparison; algorithm; usability evaluation; evaluation-integrated development

I. INTRODUCTION

Although Algorithm is the core topic of Computer Science (CS) field, not all undergraduate CS students can understand it clearly due to its complexity. Furthermore, due to limited time in class, several students cannot ask lecturer directly for help. Thus, several Algorithm Visualization (AV) have been developed to overcome these impediments [1][2][3][4], AV is an educational tool which visualize how an algorithm works in more intuitive manner [2]. With the aid of an AV, students are expected to learn and understand how algorithm works on certain problem. Students can also replay and pause animation at particular time in order to clarify their misunderstanding about certain concept.

However, CS students are not only expected to understand how an algorithm works but also why a problem should be solved by particular algorithm [5][6]. The later goal is quite difficult to achieve since students should be able to differentiate several algorithms based on their characteristics and determine which algorithm is the most suitable solution for particular problem. In order to let student understand this topic, lecturer should describe efficiency and effectiveness of each algorithm and explain why an algorithm is better than others in specific case [7]. Furthermore, based on the fact that practical approach is more easy to understand than theoritical ones, this approach is commonly implemented through case-based performance comparison. The characteristic of an algorithm is described by comparing it to a baseline algorithm on a specific input data. Unfortunately, this approach is seldom featured in most AVs.

In this paper, we extend the capabilities of AV by utilizing case-based performance comparison. Additionally, since the main goal of an AV is not only technical functionality but also its usability [8], usability evaluation should not be performed after the implementation of the system is complete [9]. Therefore, several usability evaluations are also integrated in our AV development instead of a standard blackbox testing. We integrate heuristic evaluation, query technique, controlled experiment, and observational study. For our implementation, we implement these aspects to algorithm for solving classic problems such as 0/1 knapsack and Minimum Spanning Tree (MST) problem [10]. 0/1 knapsack is solved using brute force, greedy algorithms (greedy by weight, profit, and density), backtracking, and dynamic programming whereas MST is solved using brute force, Prim, and Kruskal.

II. RELATED WORKS

For recent decades, researchers had found that visualization may aid learner to get better understanding about certain concept (e.g. data trend [11], hierarchical data [12], software [8], and even for educational materials [2][4][13][14][15]). In CS education field, visualization is commonly used for explaining how certain process or an algorithm works. This kind of tool is commonly called Algorithm Visualization (AV). Nowadays, most AVs are listed and collected in AV portals like AlgoViz [16] and VisuAlgo [17]. Both of them are intended to be one-stop solutions for AV and only differ in AV contribution. AlgoViz works as third party application which enables all AV developer to enlist their AV on their site whereas VisuAlgo creates their own AV. Although web-based AVs are popular, several AVs are still developed in desktop platform (e.g. AP-ASD1 [13]) since several cases require high computational

capabilities and not all areas are featured with fast internet access.

In spite of there are many avalaible AVs, most of them are only intended to visualize algorithm steps without mentioning why that algorithm should be chosen based on a given problem. Velázquez-Iturbide & Pérez-Carrasco state that case-based performance comparison may improve student understanding further about why an algorithm should be chosen [5]. They have developed GreedEx, which is a tool to aid student in learning greedy algorithm. By using this tool, learners can explore several greedy algorithms prepared by lecturer, compare their outputs, and conclude their algorithm characteristics. They also extend their GreedEx to GreedExCol which involve Computer-Supportive Collaborative System (CSCL) [18]. However, their tool is only focused on case-based performance comparison without algorithm visualization.

In this paper, we combine both algorithm visualization with case-based performance comparison in order to improve student knowledge further. Consequently, several supplementary features are also required to support that combination. These supplementary features are input generator, file conversion, and language preferences. Input generator is utilized since certain algorithm may require large-sized input data which is quite difficult to be created manually. File conversion aims to data portability which enables student to transfer the input/process/output to another student. Lastly, language preferences is intended to remove language barrier since our implementation AV is developed for Indonesian undergraduate students. To achieve better effectiveness, several usability evaluations are also integrated in our AV development such as heuristic evaluation, query technique, controlled experiment, and observational study. Our implementation AV is named AP-SA which focus on visualizing algorithm for solving 0/1 knapsack and Minimum Spanning Tree (MST) problem. Since performance comparison requires high computational capabilities, AP-SA is developed in desktop platform using C#.

III. Design and Implementation

Because of usability evaluation should not be performed after the implementation of the system is complete [9], design and development of our AV is modified and the phases can be seen in Figure 1. In the first phase, early design is evaluated using heuristic evaluation where all core features are analyzed and inspected. In the second phase, revised design is implemented and evaluated using query technique in order to signify missing features from student's perspective. Finally, the second attempt of implementation is evaluated using more sophisticated evaluation such as controlled experiment and observational study. Their evaluation results are analyzed and integrated as needed to yield our final AV implementation. Additionally, a black box testing is always conducted for each implementation phase on design and development.

A. Design and 1st Implementation

For initial design, we enlist all required features in order to combine algorithm visualization and case-based performance comparison. Then, all required features are analyzed and inspected through heuristic evaluation [19]. Typically, heuristic evaluation is conducted by second and third author of this paper since both of us are algorithm lecturers. Based on heuristic evaluation, several core features have been defined which are:

a) *Solving visualization:* For each algorithm in each problem, its solving mechanism should be visualized and explained step-by-step. This feature allows students to learn about how certain algorithm works in certain problem. As we know, this is a common feature for an AV.

b) *Performance comparison:* For each input data set in each problem, several algorithms can be compared based on optimality, completeness, time complexity, execution time, and output. With this feature, student can compare the characteristics of each algorithm, especially for certain problem. However, memory complexity is ignored in our performance comparison since memory usage cannot be determined due the impact of garbage collector in our development programming language (C#).

c) *Input generator:* Since input data size for each problem may be large, input generator is needed. With input generator, students can simply start learning without wasting too much time for preparing input.

d) *File conversion:* For each algorithm in each problem, its input, process, and output can be exported from or imported to raw text or CSV file. This feature aims to data portability which enable student to transfer the input/process/output to another student.

e) *Language preferences:* Since we develop this AV for Indonesian undergraduate students, this AV should provide two languages which are English and Indonesia. Students can choose which language that fits their necessity.

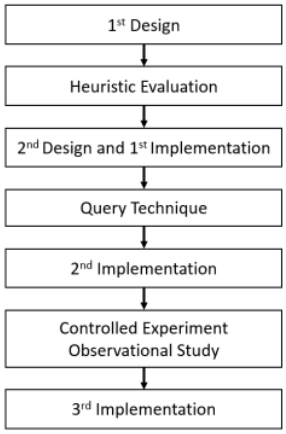

Figure 1. AP-SA Design and Implementation Phases

Besides core features, several must-have features which are grounded from best practices [1] are also embedded in our AV. These features are visualization legend, information visibility setting, performance information, execution history, flexible

execution control, learner-built visualization, customizable input data sets, and visualization-oriented explanations. Based on empirical evaluation, these features are purposed to improve student understanding further.

In order to improve the effectiveness of AP-SA, we also integrate four kinds of AV engagement which is based on AV engagement taxonomy [1]. These AV engagements with its AP-SA correlated features can be seen in Table I. Viewing is represented as step-by-step animation in problem-related solving visualization where student can see how certain algorithm works on certain problem. Changing is implemented by providing dynamic input where students can give their artificial input by hand, generating input through input generator, or importing input from file. Constructing is provided by letting students determine how the algorithm is visualized through visualization setting. Students can determine node numbering (alphabet, number, or Roman numeral), input edge color, and the visibility of edge weights. Lastly, Presenting can be conducted by letting student to present algorithm-related material with the aid of step-by-step problem-related solving visualization and performance comparison.

TABLE I. ENGAGEMENT-FEATURE CORRELATION

Engagement Form	AP-SA Correlated Feature
Viewing	Step-by-step animation in problem-related solving visualization
Changing	Dynamic input
Constructing	Visualization setting
Presenting	Step-by-step problem-related solving visualization and performance comparison

For visual representation, each algorithm in AP-SA have different visualization which details can be seen in Table II. All 0/1 knapsack problem solving except backtracking utilize dynamic table as its visualization. Dynamic table is a table which size may be changed dynamically based on its content. On the other hand, 0/1 knapsack with backtracking is visualized using search space tree since the idea of backtracking is rooted from search space concept. MST is visualized using logical graph in order to adopt the natural behavior of MST input.

TABLE II. ALGORITHMIC STRATEGIES VISUAL REPRESENTATION

Problem Solving Strategy	Visual Representation
0/1 knapsack with brute force	Dynamic table
0/1 knapsack with greedy algorithm	Dynamic table
0/1 knapsack with backtracking	Search space tree
0/1 knapsack with dynamic programming	Dynamic table
MST with brute force	Logical graph
MST with Prim	Logical graph
MST with Kruskal	Logical graph

B. *Query Technique and 2nd Implementation*

After 1st Implementation, our AV is evaluated using query technique where several opinions about different aspects of the system are collected through questionnaires [9]. Query technique is conducted to 10 undergraduate students which have known 0/1 knapsack and MST. Students are asked to give feedbacks about AP-SA core features which are solving visualization, performance comparison, file conversion, input generator, and language preferences. Based on respondent feedbacks, most revision are considered minor since it does not affect our major features directly. These revisions are color changing, button position, textual representation, and video-like controller. Then, these feedbacks are evaluated and most of them are implemented in our AV.

C. *Controlled Experiment and Observational Study*

Controlled experiment is a kind of usability evaluation which asks learners to complete tasks given by lecturer [9]. Each task is conducted to provide important information such as effectiveness, efficiency, ease of use, and other interesting issues. On the other hand, Observational study is quite similar to query technique except the way information is collected [9]. In observational study, lecturer observes how students use the system and write down every important issue. Both controlled experiment and observational study are conducted simultaneously which schedule details can be seen in Table III. This evaluation involves 13 undergraduate students which have known 0/1 knapsack and MST wherein respondents are not informed before about how this evaluation works, so that they cannot prepare anything. Furthermore, to encourage students for doing their best, prizes are also given to students with good grade on each task.

TABLE III. CONTROLLED EXPERIMENT AND OBSERVATIONAL STUDY TASK SCHEDULES

Task	Time-on-task (minutes)
Pre-test (6 questions)	10
Imitating tutor scenario (9 scenarios)	40
Application-based problem solving (140 short-answer questions)	30
Questionnaire survey (11 questions)	10
Post-test (6 questions)	10

Pre-test, post-test, and questionnaire survey are utilized to measure AV effectiveness. Pre-test and post-test measure AV effectiveness from student perspective, defining how far an AV can improve student knowledge. Generally, post-test should yield greater result than pre-test since respondent have learned something from AV. On the other hand, questionnaire survey is conducted to measure each effectiveness aspects in detail. Respondents are given several statements which are required to be graded based on their perspective.

Pre-test and post-test consist of 6 questions where each question is related to certain level in Bloom taxonomy [20]. Although both test have similar questions, they differ in execution timing. Pre-test is conducted before other tasks whereas post-test is conducted after them. Pre- and post- test question detail can be seen in Table IV. Knowledge-level question is the easiest one since the answer of this question can be found directly on literature whereas Evaluation-level question is the hardest since student need to know the detail and

implementation of Prim and Kruskal algorithm, especially for solving MST problem.

TABLE IV. PRE- AND POST-TEST QUESTION LIST

Question Level	Question Detail
Knowledge	Ask students to determine brute force characteristic
Comprehension	Ask students to arrange algorithm steps for solving MST using Kruskal's algorithm
Application	Ask students to solve 0/1 knapsack using greedy algorithm
Analysis	Ask students to choose the most efficient algorithm to solve 0/1 knapsack
Synthesis	Ask students to provide input for 0/1 knapsack so that greedy by weight may yield optimum result
Evaluation	Ask students to choose Prim or Kruskal to solve MST. Their answer should be featured with reasonable argument.

Pre-test and post-test result can be seen in Table V wherein improvement is represented as percentage of improvement based on pre-test score. Each question are graded from 0 to 1 inclusively so that overall score for each test is ranged from 0 to 6 inclusively. As seen in Table IV, post-test result is always greater than pre-test result from both single-question and overall perspective. Additionally, the improvement between overall scores is quite high (73,540%). Thus, it concludes that our AV is quite effective in terms of improving student knowledge. For Application-level question, its improvement is relatively small since most respondents pay less attention to small details in solving visualization.

TABLE V. PRE- AND POST-TEST RESULT

Question Level	Average Score		
	Pre-test	Post-test	Improvement (%)
Knowledge	0,307	0,692	125,000
Comprehension	0,115	0,615	433,333
Application	0,653	0,661	1,176
Analysis	0,461	0,538	16,667
Synthesis	0,631	0,938	48,780
Evaluation	0,069	0,438	533,333
Overall Score (Sum)	2,238	3,884	73,540
Overall Score (Average)	0,373	0,647	73,540

Questionnaire survey is conducted by asking respondents to grade several AP-SA functionality-related statements based on their perspective. For each statement, respondent should give an integer ranged from 1 to 5 inclusively where 1 means *very disagree*, 2 means *disagree*, 3 means *neutral*, 4 means *agree*, and 5 means *very agree*. This survey consists of 11 statements which are 2 functionality statements, 1 intuitiveness statement, 1 consistency statement, 1 concept statement, 1 terminology statement, and 5 core feature statements. As seen in Table VI, 8 of 11 statements yield result greater than 4 which conclude that these statements are agreed by respondents. However, 3 of them yield result lower than 4 (though it still higher than 3). These statements are consistency, file conversion, and language preference statements. File conversion statement is rated lower than 4 since input file generated from our AV is in CSV format which may not descriptive enough for some respondents. Both consistency and language preference statements are rated lower than 4 due to inconsistent language translation on AP-SA during evaluation. We also ask students to write down any feature-related feedback about AP-SA. These feedbacks are categorized as follows, inconsistent language translation, UI look and feel, and more-simple input representation. Inconsistent language translation occurs since several terms are still not translated. However, all inconsistent translation have been listed and corrected in final implementation of AP-SA. UI look and feel feedback is resulted since our AV utilize white as its background color (which is too bright for a respondent). Input representation feedback is resulted since AP-SA utilize adjacency matrix as input for MST problem. Both feedbacks will be implemented in next research since both of them require considerable effort and occurs after major implementation (2nd implementation).

TABLE VI. SURVEY STATISTICS

Statement	Average Score
AP-SA functionality may aid learner to learn algorithm for solving 0/1 knapsack	4,615
AP-SA functionality may aid learner to learn algorithm for solving MST	4,692
AP-SA has intuitive UI	4,076
Layout and functionality of AP-SA are consistent	3,384
AP-SA is effective to learn the concept of algorithm	4,461
Terminology and materials are commonly used in undergraduate course	4,692
Step-by-step visualization may enchance learner knowledge of algorithm to solve specific problem	4,769
Performance comparison may help learner to differentiate several algorithm	4,153
Input generator may simplify learning since students are not required to wasting time to create input by hand. As we know, input size may be large.	4,692
File conversion may aid learner in terms of data portability	3,923
Language preference may aid learner to learn in their native language	3,461
Average Score	4,265

Based on survey result, the impact of core features are sorted descending as follows: solving visualization, input generator, performance comparison, file conversion, and language preferences. Solving visualization get the highest score since it is considered as the core aspect of an AV. Although input generator is a supplementary feature for performance comparison, it still yield higher score than performance comparison since giving input is a compulsory step before conducting other features. File conversion and language preferences yield low result among core features since they are not directly related to algorithm.

Imitating tutor scenario and application-based problem solving are utilized to measure AV ease of use and to observe student behaviour. Besides, both tasks are also utilized to introduce AP-SA features. Imitating tutor scenario aims to introduce it at Knowledge and Comprehension level whereas

application-based problem solving introduce it at higher level. By introducing AP-SA features, students are expected to give more objective results on questionnaire survey.

For imitating tutor scenario, students are asked to reproduce similar scenario as described by tutor before. This experiment consists of 7 scenarios which are based on AP-SA core features (2 solving visualization scenarios, 2 performance comparison scenarios, 1 input generator scenario, 1 file conversion scenario, and 1 language preference scenario). Since student completion time for each task in imitating tutor scenario is gradually shortened, it can be concluded that several respondents require a few minutes to adapt with AP-SA, but they can use it well at the rest of the time.

On the other hand, application-based problem solving is conducted by asking students to answer 140 simple questions which classified to 9 categories, 4 for MST problems (brute force, Prim, Kruskal, and algorithm comparison) and 5 for 0/1 knapsack problems (brute force, greedy algorithm, backtracking, dynamic programming, and algorithm comparison). During this test, most students can answer all questions easily. Furthermore, they also can complete it faster than allocated time. However, to check whether students answer it seriously or not, we also grade their answer which averaged result is 93,384%. Since this averaged result is quite high, we can conclude that students answer it seriously. Due to the fact that students can complete their task faster than expected, it can be concluded that AP-SA is quite easy to use. This conclusion is also deducted from the result of functionality and intuitiveness statements on survey.

Based on both evaluations, we also collect several student comments which are stated when they are doing the tasks. However, all comments are similar with their feedbacks on questionnaire which requires no additional analysis in this section.

D. 3rd Implementation

The main window of AP-SA can be seen in Figure 2 and adapted from our previous AV, AP-ASD1 [13]. It consists of several components such as title panel (A), module selection (B), input panel (C), visualization panel (D), and visualization explanation panel (E). They are represented in Figure 2 as A to E respectively. Title panel consists of several features such as title, video-like animation controller, language selection, algorithmic strategies problem explanation, and tutorial. Students can select which module they wish to learn from module selection. After selecting a module, students can give, generate, or import certain input and configure visualization setting in input panel. Finally, students can start to learn through animation visualized in visualization panel with its explanation showed in visualization explanation panel.

The example of problem-solving visualization using AP-SA can be seen in Figure 3 and Figure 4. Figure 3 represents problem-solving visualization of 0/1 knapsack problem with backtracking whereas Figure 4 represents problem-solving visualization of MST with Prim's algorithm. As seen in Figure 3 and Figure 4, visualization explanation panel is split into two sub-panels where left panel represents textual explanation and the right one represents the best solution so far.

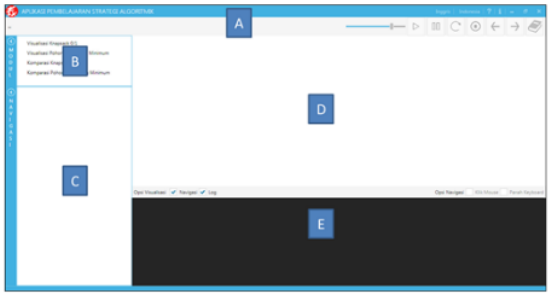

Figure 2. AP-SA Main Window

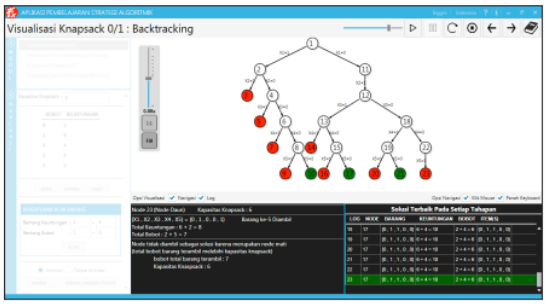

Figure 3. Problem-solving Visualization of 0/1 Knapsack Problem with Backtracking

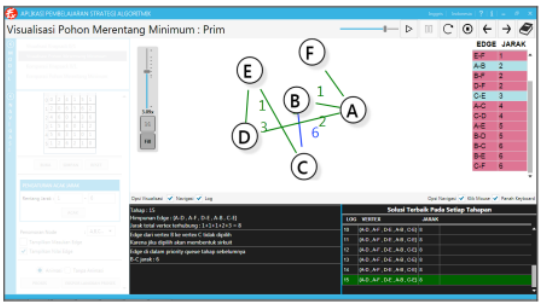

Figure 4. Problem-solving Visualization of MST with Prim's Algorithm

The example of performance comparison using AP-SA can be seen in Figure 5 which represents the result of comparing several algorithms for solving 0/1 knapsack problem. Time elapsed and the result for each algorithms are shown in visualization panel whereas its supplementary information (time complexity, optimality, and completeness) can be seen in visualization explanation panel.

IV. CONCLUSIONS

Based on our research, several conclusions can be stated which are:

a) Performance comparison in AV may enchance student knowledge about algorithm. This statement is concluded from the result of performance comparison

statement in questionnaire which yields fairly good result (4,153 of 5).

b) Integrating usability evaluation in AV design and development may enchance AV's impact since the main goal of an AV is user-oriented and requires many feedbacks from users. Feedbacks can be collected either implicitly (e.g. pre-test and post-test, imitating tutor scenario, and application-based problem solving) or explicitly (e.g. query technique and questionnaire survey).

c) Based on controlled experiment and observational study, it can be concluded that our implementation AV, AP-SA is easy to use and quite effective to improve student knowledge about algorithm. Additionally, our core features also fit student need based on survey result.

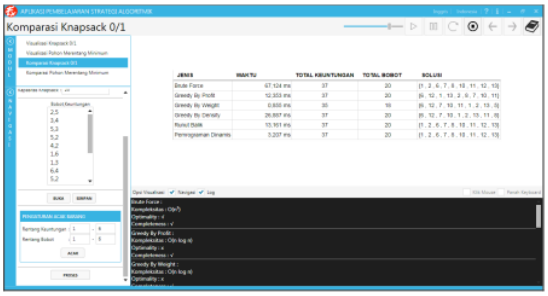

Figure 5. Performance Comparison in 0/1 Knapsack Problem

V. FUTURE WORK

For further research, we intend to measure the impact of AV engagement described by Naps and determine which engagement may fit undergraduate students in Indonesia. Furthermore, we also intend to determine the most understandable visual representation for AV. From implementation AV perspective, we will integrate UI look and feel and input selection in further research.

REFERENCES

[1] T. L. Naps, G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger and J. A. Velázquez-Iturbide, "Exploring the role of visualization and engagement in computer science education," in *ITiCSE-WGR '02 Working group reports from ITiCSE on Innovation and technology in computer science education*, New York, 2003.

[2] 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 (TOCE),* vol. 10, no. 3, 2010.

[3] E. Fouh, M. Akbar and C. A. Shaffer, "The role of visualization in computer science education," *Computers in the Schools: Interdisciplinary Journal of Practice, Theory, and Applied Research,* vol. 29, no. 1-2, pp. 95-117, 2012.

[4] S. Halim, Z. C. Koh, V. B. H. Loh and F. Halim, "Learning Algorithms with Unified and Interactive Web-Based Visualization," *Olympiads in Informatics,* vol. 6, pp. 53-68, 2012.

[5] J. Á. Velázquez-Iturbide and A. Pérez-Carrasco, "Active learning of greedy algorithms by means of interactive experimentation," in *ITiCSE '09 Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education*, New York, 2009.

[6] "Curriculum Guideliness for Undergraduate Degree Programs in Computer Science," ACM and IEEE Computer Society. The Joint Task Force on Computing Curricula: Computer Science Curricula 2013, 2013. [Online]. Available: http://www.acm.org/education/CS2013-final-report-pdf.

[7] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach (3rd Edition), Prentice Hall, 2009.

[8] C. D. Hundhausen, S. A. Douglas and J. T. Stasko, "A Meta-Study of Algorithm Visualization Effectiveness," *Journal of Visual Languages & Computing,* vol. 13, no. 3, p. 259–290, 2002.

[9] O. Kulyk, R. Kosara, J. Urquiza and I. Wassink, "Human-Centered Aspects," in *Human-Centered Visualization Environments*, Springer-Verlag, 2007, pp. 13-75.

[10] R. Sedgewick and K. Wayne, Algorithms (4th Edition), Princeton, 2011.

[11] G. Robertson, R. Fernandez, D. Fisher, B. Lee and J. Stasko, "Effectiveness of Animation in Trend Visualization," *IEEE Transactions on Visualization and Computer Graphics,* vol. 14, no. 6, pp. 1325 - 1332, 2008.

[12] Y. Tu and H.-W. Shen, "Visualizing Changes of Hierarchical Data using Treemaps," *IEEE Transactions on Visualization and Computer Graphics,* vol. 13, no. 6, pp. 1286 - 1293, 2007.

[13] L. Christiawan and O. Karnalim, "AP-ASD1 An Indonesian Desktop-based Educational Tool for Basic Data Structures," *Jurnal Teknik Informatika dan Sistem Informasi (JuTISI),* vol. 2, no. 1, 2016.

[14] T. L. Naps, "JHAVE: Supporting algorithm visualization," *IEEE on Computer Graphics and Applications,* vol. 25, no. 5, pp. 49-55, 2005.

[15] V. Karavirta and C. A. Shaffer, "JSAV: the JavaScript algorithm visualization library," in *The 18th ACM conference on Innovation and technology in computer science education*, New York, 2013.

[16] "AlgoViz.org : The Algorithm Visualization Portal," [Online]. Available: http://algoviz.org/. [Accessed 7 12 2015].

[17] S. Halim, "VisuAlgo," [Online]. Available: http://visualgo.net/. [Accessed 12 5 2015].

[18] O. Debdi, M. Paredes-Velasco and J. Á. Velázquez-Iturbide, "GreedExCol, A CSCL tool for experimenting with greedy algorithms," *Computer Applications in Engineering Education,* vol. 23, no. 5, pp. 790-804, 2015.

[19] J. Urquiza-Fuentes and J. Á. Velázquez-Iturbide, "A Survey of Successful Evaluations of Program Visualization and Algorithm Animation Systems," *ACM Transactions on Computing Education (TOCE) - Special Issue on the 5th Program Visualization Workshop (PVW'08),* vol. 9, no. 2, 2009.

[20] B. Bloom and D. Krathwohl, Taxonomy of Educational Objectives : the Classification of Educational Goals Handbook I: Cognitive Domain, Addison Wesley, 1956.

Extending The Effectiveness of Algorithm Visualization with Performance Comparison through Evaluation-integrated Development

ORIGINALITY REPORT

3%	2%	3%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to School of Business and Management ITB Student Paper	2%
2	Submitted to iGroup Student Paper	1%
3	Jaime Urquiza-Fuentes, J. Ángel Velázquez-Iturbide. "A Survey of Successful Evaluations of Program Visualization and Algorithm Animation Systems", ACM Transactions on Computing Education, 2009 Publication	1%

Exclude quotes	Off	Exclude matches	< 1%
Exclude bibliography	On		