Introducing an Educational Tool for Learning Branch & Bound Strategy

Sofriesilero Zumaytis[1)], Oscar Karnalim[2)]

[1)2)]*Faculty of Information Technology, Maranatha Christian University*
Jl.Prof.drg.Suria Sumantri, MPH No. 65, Bandung, Indonesia

[1)]sofriesilero.zumaytis@gmail.com

[2)]oscar.karnalim@it.maranatha.edu

Abstract—**According to our informal survey, Branch & Bound strategy is considerably difficult to learn compared to other strategies. This strategy consists of several complex algorithmic steps such as Reduced Cost Matrix (RCM) calculation and Breadth First Search. Thus, to help students understanding this strategy, AP-BB, an educational tool for learning Branch & Bound is developed. This tool includes four modules which are Brute Force solving visualization, Branch & Bound solving visualization, RCM calculator, and case-based performance comparison. These modules are expected to enhance student's understanding about Branch & Bound strategy and its characteristics. Furthermore, our work incorporates TSP as its case study and Brute Force strategy as a baseline to provide a concrete impact of Branch & Bound strategy. According to our qualitative evaluation, AP-BB and all of its features fulfil student necessities for learning Branch & Bound strategy.**

Keywords— **Educational Tool, Branch & Bound, Algorithm Strategy, Algorithm Visualization**

Article history:
Received 10 January 2017; Received in revised form 23 February 2017; Accepted 23 February 2017;
Available online 28 April 2017

I. INTRODUCTION

Despite the fact that Algorithm is the core topic of Computer Science (CS), not all undergraduate CS students can understand it properly by only relying on in-class session. Several students may require more time to understand it than others whereas some of them may require more detailed explanation than others. Therefore, to handle this issue, several algorithm-centric CS educational tools are developed. Using these tools, students can learn algorithm independently without relying heavily on in-class session.

According to the fact that CS students are not only expected to understand how an algorithm works but also why an algorithm is better than others (Velázquez-Iturbide & Pérez-Carrasco, 2009; Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society, 2013), several CS educational tools are focused on comparing algorithms and exploiting algorithm characteristics. GreedEx (Velázquez-Iturbide & Pérez-Carrasco, 2009), GreedExCol(Debdi, Paredes-Velasco, & Velázquez-Iturbide, 2015), AP-SA (Jonathan, Karnalim, & Ayub, 2016), and Complexitor (Elvina & Karnalim, in press) are several tools which fall into this category. Using these tools, students are encouraged to observe and differentiate various algorithms through their respective characteristics.

Branch & Bound is an algorithm strategy for solving state-space-search-based problem by incorporating heuristic and Breadth-First-Search (BFS) (Levitin, 2012). This strategy is frequently incorporated on various popular tasks such as Travelling Salesperson Problem (TSP) and N-Queens. However, according to our informal survey, this strategy is quite difficult to be understood by undergraduate students. Most students are overwhelmed by numerous steps required to solve a given problem. In addition, some of them also lack of visual imagination which makes them harder to visualize Branch & Bound logical tree for solving a problem.

To overcome this issue, this paper proposes AP-BB, a CS educational tool for learning Branch & Bound with TSP as its case study. This tool is intended to aid student for learning Branch & Bound strategy and its characteristics. Branch & Bound algorithmic's steps are covered by solving visualization whereas its characteristics are covered by case-based performance comparison. Solving visualization is split into three modules which are Brute Force solving visualization, Branch & Bound solving visualization, and Reduced Cost Matrix (RCM) calculator. The first module is responsible to cover Brute Force strategy whereas the latter two are responsible to cover Branch & Bound strategy. Brute Force is incorporated in our AP-BB module as a baseline for exploiting Branch & Bound strategy. Both strategies are compared based on their respective characteristics on case-based performance comparison module.

II. RELATED WORKS

Algorithm is a Computer Science (CS) topic which is mandatory understandable by CS

e-ISSN 2443-2555
doi: http://dx.doi.org/10.20473/jisebi.3.1.8-15

students. However, due to various cognitive skills among CS students, in-class session may be not sufficient to fulfill all students learning process, especially for the weak ones. Thus, to overcome this issue, several CS educational tools for learning algorithm are developed. These tools are expected to help students learn algorithm-related topic beyond in-class session. Students can learn a particular topic by themselves without depending on in-class lecture. For teaching algorithm, CS educational tools may target various level of algorithm understanding. It starts from technical implementation (e.g., program creation) to abstractive form (e.g., algorithmic steps).

CS educational tools which focused on technical implementation are frequently referred as Program Visualization (PV) (Bentrad & Meslati, 2011). These tools encourage students to implement their algorithm into source codes and assist them to understand it properly (Guo, 2013; Rajala, Laakso, Kaila, & Salakoski, 2008; Cisar, Pinter, Radosav, & Cisar, 2010).Python Tutor (Guo, 2013), Jeliot 3 (Cisar, Pinter, Radosav, & Cisar, 2010), JIVE (Gestwicki & Jayaraman, 2002), and VILLE (Rajala, Laakso, Kaila, & Salakoski, 2008)are several examples that fall into this category. These tools visualize all variables and function calls from given source code as its execution advances step-by-step. In such manner, students can analyze and learn how their algorithm works directly through its implementation (i.e. source code). In addition, some of them also incorporate several supplementary engagements such as pop-up question to enhance student understanding.

According to the fact that several students are overwhelmed by a large amount of compile errors when developing algorithm implementation (i.e. source code), a tool named Verificator (Radosevic, Orehovacki, & Lovrencic, 2009)incorporates a "traffic-light" system to limit the number of compile errors. Students are forced to compile their code every N modifications and they can only continue to write code *if* their code is successfully compiled. In such manner, students will experience only a small amount of errors each time a source code being compiled.

However, since limiting compile errors does not mean removing them completely, several tools replace conventional source-code-writing task with visual representation to avoid syntactic errors. Students are encouraged to drag-and-drop several components to generate source code (i.e. algorithm implementation). Each time a component is added, students should provide necessary information for given component in limited input fields. In such manner, no syntactic errors will be produced at compile phase since all potential errors have already handled directly when adding each component. ProGuide (Areias & Mendes, 2007), Raptor (Carlisle, Wilson, Humphries, & Hadfield, 2005), SFC editor (Watts, 2004), and Alice (Cooper, Dann, & Pausch, 2000) are several examples that fall into this category. Among these tools, Alice is the only work which incorporates fun as their major contribution. It enables student to learn algorithm implementation through interactive environments (e.g. 3D visualization and real-world object).Students can arrange their own algorithm based on provided instructions and see how their composed algorithm works in interactive manner. This kind of approach is inspired from other fun-based CS educational tool such as CeeBot4 (Anonymous, Learn Programming with Ceebot4, 2008) and KarelRobot (Buck & Stucki, 2001),

In more abstractive manner, several CS educational tools are focused on teaching algorithm in the form of algorithmic steps. Students are encouraged to understand algorithm by learning how that algorithm works step by step. These tools are frequently referred as Algorithm Visualization (AV) tools since they incorporate visualization as its main engagement form (Shaffer, et al., 2010; Halim, Koh, Loh, & Halim, 2012; Christiawan & Karnalim, 2016; Jonathan, Karnalim, & Ayub, 2016). Nowadays, most AV tools are listed in AV portals such as AlgoViz (Anonymous, AlgoViz.org : The Algorithm Visualization Portal, 2009) and VisuAlgo (Halim, VisuAlgo, 2010). Students can access these portals through the Internet and utilize them to learn algorithms directly. However, due to varied university course need, several AV tools are also re-developed locally to match course syllabus (Christiawan & Karnalim, 2016; Jonathan, Karnalim, & Ayub, 2016). These tools frequently outperform standard AV tools in terms of its effectiveness due to its high-synchronization with course materials.

Even though there are various algorithm-centric CS educational tools, only a few of them that is focused on algorithm characteristics. Most of them are only focused on applying algorithm to solve a particular problem. GreedEx (Velázquez-Iturbide & Pérez-Carrasco, 2009), GreedExCol(Debdi, Paredes-Velasco, & Velázquez-Iturbide, 2015), AP-SA (Jonathan, Karnalim, & Ayub, 2016), and Complexitor (Elvina & Karnalim, in press) are several AV tools which are focused on algorithm characteristics. Using algorithm characteristics, students are expected to determine why an algorithm is better than others for solving given problem.

GreedEx is an educational tool for learning Greedy Algorithm characteristics in comparative manner. Students can explore how several greedy algorithms work, compare their respective output, and determine which greedy algorithm is the best approach for solving the given problem.

GreedExCol is an extended-version of GreedEx. It incorporates Computer-Supportive Collaborative System (CSCL), so that students can share and argue about their work in collaborative manner.

AP-SA (Jonathan, Karnalim, & Ayub, 2016) is an AV tool for teaching various algorithm strategies. It incorporates case-based performance comparison as one of their core features, so that students can compare the characteristics of each algorithm and determine which algorithm is the best to solve the given problem. It incorporates 4 algorithm strategies which are Brute Force, Greedy Algorithm, Backtracking, and Dynamic Programming. These strategies are implemented by involving two case studies which are 0/1 Knapsack and Minimum Spanning Tree (MST) 0/1.

Complexitor (Elvina & Karnalim, in press) is an educational tool for learning algorithm time complexity in practical manner. This tool, at some extent, may enhance student understanding for selecting the best algorithm in terms of its time complexity. Students can calculate time complexity based on algorithm implementation (i.e. source code) and input set. Complexitor incorporates two measurements for calculating time complexity. These measurements are actual processing time and the number of involved instructions. The number of involved instructions is incorporated to calculate time complexity based on small input set. As we know, actual processing time may yield biased result on small input set due to hardware and operating system dependency.

To our knowledge, there are no algorithm-centric CS educational tool which covers Branch & Bound strategy and its characteristics. Branch & Bound is an optimization technique for solving state-space-search-based problem by incorporating heuristic and Breadth-First-Search (BFS) (Levitin, 2012). It expands current branches in BFS manner till a feasible solution found and bounds all possible solution with higher heuristic than the feasible one. Even though this strategy is less popular than Greedy or Dynamic Programming (DP) strategy, it is still incorporated in various tasks such as Travelling Salesperson Problem (TSP) and N-Queens. Therefore, this strategy is considerably important to be understood by CS students.

This paper proposes an educational tool for learning Branch & Bound through Visualization. It is named AP-BB which is an acronym of "educational tool for learning Branch & Bound" in Indonesian language. AP-BB is developed based on the fact that Branch & Bound strategy is quite difficult to learn among all algorithm strategies taught on Information Technology major in our university. This finding is deducted based on our informal observation of student tests on Algorithmic Strategy course from academic year of 2014/2015 and 2015/2016. Most students fail to answer Branch & Bound problem correctly.

In order to provide a concrete example for representing the impact of Branch & Bound strategy, our work incorporates TSP as its case study. TSP is a Non-Polynomial problem which asks the shortest Hamilton circuit from given cities based on its distance (Levitin, 2012). Hamilton circuit refers to a cycle traversing all nodes in graph once (i.e. cities) that starts and ends on the same city. Furthermore, our work incorporates Brute Force as a baseline to draw out Branch & Bound characteristics. Both strategies are implemented for solving similar problem (TSP) so that they are comparable to each other.

In general, AP-BB consists of four modules which are Brute Force solving visualization, Branch & Bound solving visualization, Reduced Cost Matrix (RCM) calculator, and case-based performance comparison. The first two modules are incorporated to learn TSP problem solving with Brute Force or Branch & Bound strategy; RCM calculator is incorporated to learn RCM calculation as Branch & Bound TSP heuristics; and Case-based performance comparison is incorporated to analyze Branch & Bound characteristic when solving a particular TSP case. These characteristics are expected to enhance student understanding further about Branch & Bound strategy and its characteristics.

III. AP-BB DESIGN

(Jonathan, Karnalim, & Ayub, 2016) proposes five major features for learning algorithm strategy. These features are solving visualization, case-based performance comparison, file conversion, input generator, and language preference. Among these features, AP-BB only incorporates the first three features. Input generator and language preference are excluded based on following reasons: 1) Input generator is not required in AP-BB since our input set is preferably small. We limit our input set into TSP with 6 cities to keep the clarity of our visualization and solving steps. As we know, Branch & Bound strategy is quite complex and difficult to understand, especially for solving large input set; 2) Language preference is not required in AP-BB since our tool is developed using student's native language (i.e. Indonesian language). Thus, there will be no language barrier on student learning process.

Solving visualization in AP-BB is split into three modules which are Brute Force solving visualization (BFSV), Branch & Bound solving visualization (BBSV), and Reduced Cost Matrix calculator (RCMC). BFSV and BBSV are incorporated for learning TSP problem solving with Brute Force or Branch & Bound strategy respectively. Whereas RCMC is incorporated for learning RCM calculation which is required for

Branch & Bound TSP problem solving. Even though RCM calculation is actually a part of Branch & Bound TSP problem solving, its module (RCMC) is separated with BBSV to simplify BBSV algorithmic steps. By assuming that students have already known how to calculate RCM, BBSV can consider RCM calculation as a single algorithmic step. In other words, it may reduce visualized algorithmic steps and simplify the processes. Furthermore, all of the solving visualization modules are featured with file conversion, so that students can save provided algorithmic steps on a text file. This file can be used as a reference for further discussion about given algorithmic steps with other student.

Case-based performance comparison (CPC) in AP-BB compares Branch & Bound with Brute Force strategy in terms of its performance characteristics. This module is represented as a separate module which is placed at the same level with solving visualization modules. Consequently, our AP-BB consists of four modules wherein three of them are solving visualization modules and one of them is case-based performance comparison.

Our AP-BB main window can be seen on Figure 1 and Figure 2. Figure 1 represents default view whereas Figure 2 represents its view when running a module. AP-BB main window consists of five sub-panels which are module selection, input form, control panel, visualization panel, and description panel. These sub-panels are referred as A, B, C, D, and E respectively on Figure 1. Module selection enables students to select which modules they want to learn. For each module, students can provide input set through input form and start learning through visualization given on visualization panel. However, to enhance student's understanding, each Visualization step is also featured with supplementary information which can be seen on description panel. This information is expected to help students understand the given visualization at a particular state. Control panel enables students to control the solving visualization. They can set the animation speed, replay the overall visualization, skip several visualization states, and return to previous visualization state. In addition, control panel is also featured with application tutorial, so that students can adapt and incorporate AP-BB features with ease.

A. Brute Force Solving Visualization (BFSV)

This module is incorporated to provide initial information about TSP default problem solving (i.e., Solving TSP with Brute Force). It generates all possible Hamilton paths and selects path with the lowest distance as its result. In order to synchronize our visualization with in-class lecture, this module is visualized with logical tree which sample view can be seen in Figure 3. Each node represents current city whereas each edge represents distance required from current to next city. At the end of the visualization, solution path will be marked with green color. It starts from root as its initial city and ends on a leaf node as its destination.

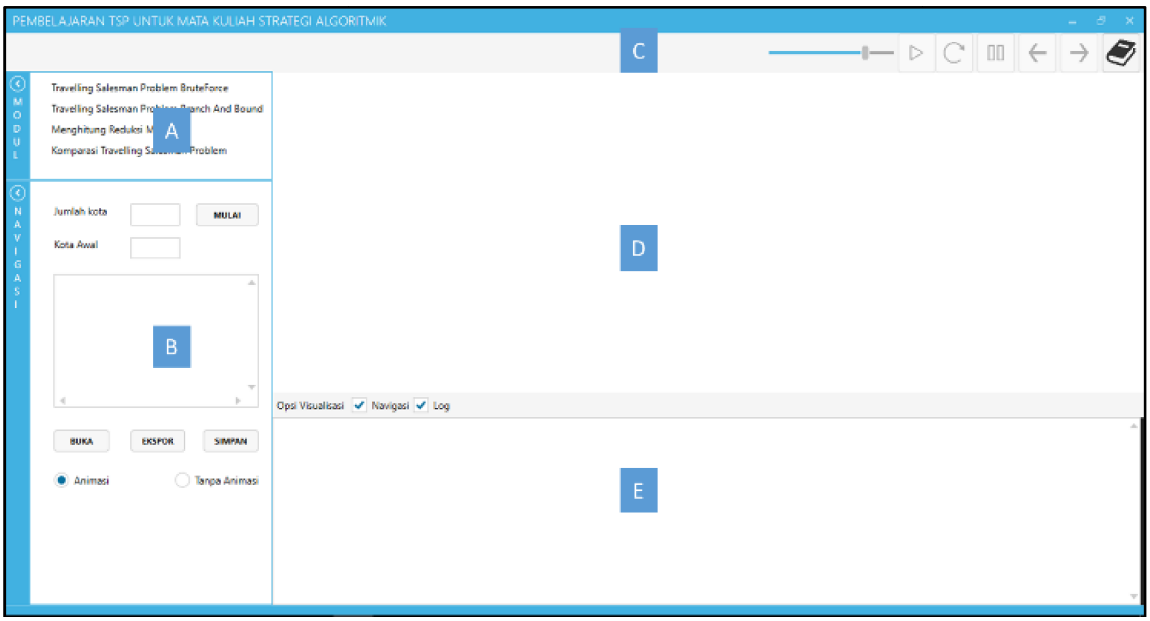

Figure 1. AP-BB Default View

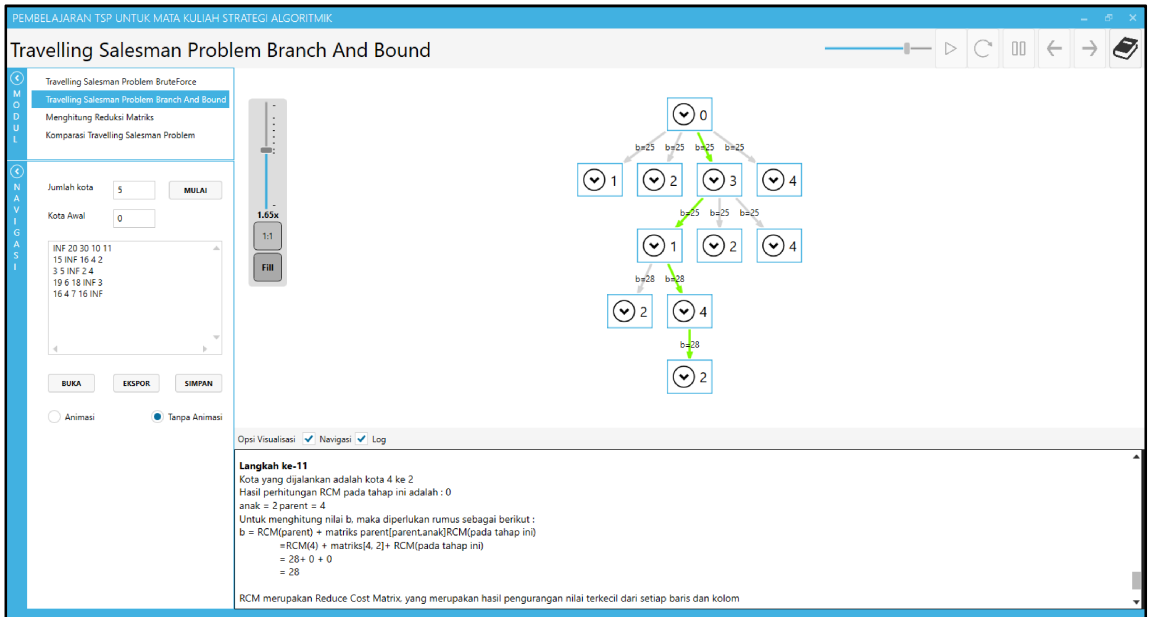

Figure 2. AP-BB View When Running a Module

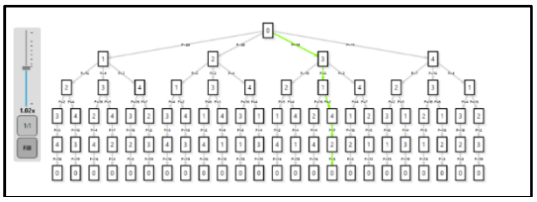

Figure 3. Brute Force Visual Representation

B. Branch & Bound Solving Visualization (BBSV)

This module is incorporated to learn Branch & Bound strategy in solving TSP. BFS traversal is incorporated till a feasible solution is found and all solution candidates with higher heuristic are automatically removed from consideration. Similar with BFSV, this module is also visualized with logical tree. Yet, logical tree in this module incorporates expand view to display RCM. Sample view of this module visual representation can be seen on Figure 4 whereas its expand view can be seen on Figure 5. Each node represents current city whereas each edge represents heuristic value required from initial to next city. For each node, students can view its RCM by clicking that node. After clicked, an expanded view will be shown displaying RCM value as seen on Figure 5.

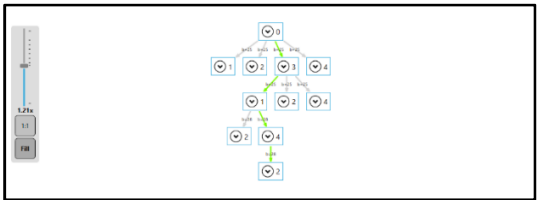

Figure 4. Branch & Bound Visual Representation

C. Reduced Cost Matrix Calculator (RCMC)

This module is incorporated to learn RCM calculation based on given matrix. RCM is generated based on two-fold which are row and column reduction (Levitin, 2012). Row reduction starts from the first row to the last one. It reduces all elements on given row with its lowest value. For example, if a row consists of ∞, 20, 30, 10, and 11, thus its reduced form will be ∞, 10, 20, 0, and 1 since each element is reduced by 10 as its lowest value. Column reduction works similar with row reduction except that it works on column instead of row. It is conducted right after row reduction is completed. Visual representation incorporated in this module is a matrix which sample view can be seen on Figure 6. Gray cells represent matrix index; White cells represent matrix content; and INF represents infinity value (∞). During visualization, selected row/column will be colored yellow and its lowest value will be colored green for each reduction.

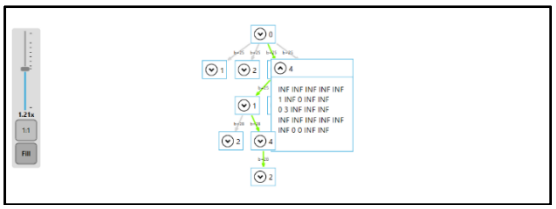

Figure 5. RCM View on Branch & Bound Visual Representation

	0	1	2	3	4
0	INF	20	30	10	11
1	15	INF	16	4	2
2	3	5	INF	2	4
3	19	6	18	INF	3
4	16	4	7	16	INF

Figure 6. RCM Calculation Visual Representation

D. Case-based Performance Comparison

This module is incorporated to learn Branch & Bound characteristic when compared with Brute Force. Based on given input set, characteristics for both strategies are recorded and displayed as standard table on visualization panel. In such manner, students can analyze and determine why Branch & Bound is better than Brute Force (i.e. default TSP solving mechanism). However, since Branch & Bound is an optimization technique to minimize execution time, characteristics incorporated in this module are limited to time-based characteristics. These characteristics are best-case time complexity, worst-case time complexity, actual processing time, and the number of involved instructions. The first two characteristics are statically similar regardless of input set. They are displayed to provide theoretical foundation about both strategies. On the other hand, the latter two characteristics are dynamically changed depending on input set. Actual processing time is measured in nanoseconds whereas the number of involved instructions is measured based on executed processes for solving given problem. The number of involved execution is displayed together with actual processing time since actual processing time may yield biased result on small input set due to hardware and operating system dependency. The number of involved execution is calculated by embedding standard counter mechanism on algorithm implementation. This mechanism is inspired from Complexitor's approach (Elvina & Karnalim, in press) for calculating time complexity in practical manner.

IV. EVALUATION

In order to prove the effectiveness of our tool, a qualitative evaluation is conducted to 20 undergraduate students from Faculty of Information Technology, Maranatha Christian University, Indonesia. Respondents are asked to answer 13 questionnaire statements on 7-points Likert scale (1 represents strongly disagree; 2 represents disagree; 3 represents negative neutral; 4 represents neutral; 5 represents positive neutral; 6 represents agree; and 7 represents strongly agree). However, to keep the validity of our evaluation, respondents are limited to students that have already taken Algorithmic Strategy course in their previous semesters. Algorithmic Strategy is a CS-based course which includes Branch & Bound strategy as one of its syllabus material. In such manner, all respondents are guaranteed to have known Branch & Bound strategy in general. This prior knowledge is required since AP-BB assumes that students have already known several algorithm-strategy terminologies such as node, edge, and heuristics.

The detail and result of our qualitative evaluation can be seen on Table I. For convenience, each statement is assigned with unique ID and will be referred as its ID at the rest of this paper. In addition to average score, each questionnaire statement is also featured with its standard deviation. Low standard deviation means that most given scores are around its average score. In other words, most respondents have similar perspective toward given statement.

Q1 and Q2 are incorporated to observe Brute Force and Branch & Bound difficulty based on student's perspective. According to Table I, our respondents tend to positively agree that Branch &Bound is difficult to understand (Q2 > 5). In addition, according to its low standard deviation, it can also be stated that most respondents have similar perspective about it. This finding indirectly strengthens our informal observation result described in Related Works. Branch & Bound is

TABEL 1. QUESTIONNAIRE STATEMENTS AND ITS RESULT

ID	Statement	Averaged Score	Standard Deviation
Q1	Brute Force strategy for solving TSP are difficult to understand	3.6	1.729
Q2	Branch & Bound strategy for solving TSP are difficult to understand	5.05	1.146
Q3	Visualization helps students to learn Brute Force strategy for solving TSP	6.05	1.191
Q4	Visualization helps students to learn Branch & Bound strategy for solving TSP	5.75	1.099
Q5	Dynamic input helps students to learn Brute Force strategy for solving TSP	5.05	1.517
Q6	Dynamic input helps students to learn Branch & Bound strategy for solving TSP	5.8	1.105
Q7	RCM calculator helps students to learn Branch & Bound strategy for solving TSP	5.95	0.826
Q8	Case-based performance comparison is effective to analyze Branch & Bound characteristics and exploit its benefits	5.8	1.056
Q9	The number of involved instructions in case-based performance comparison helps students to understand how time complexity works	5.45	1.191
Q10	AP-BB UI fulfills user necessity in terms of learning Branch & Bound strategy	6.2	1.005
Q11	AP-BB functionality fulfills user necessity in terms of learning Branch & Bound strategy	6.05	0.887
Q12	AP-BB tutorial is declarative	5.6	1.603
Q13	In general, AP-BB helps students to learn Branch & Bound strategy	6.25	0.786

quite difficult to learn when compared with other strategies. On the contrary, our respondents tend to disagree that Brute Force is also difficult to understand (Q1 < 4). This finding is natural since Brute Force is a naive approach which directly solves the problem. Thus, its algorithmic steps are simpler than Branch & Bound. Despite of its simpler algorithmic steps, Brute Force is still perceived as a difficult strategy for several students. 2 of 20 respondents are agreed that Brute Force is a difficult strategy (Q2 ≥ 6) whereas 4 of them tend to agree about its difficulty (Q2 > 5). This score variance yields high standard deviation for Q2. It yields1.729 of standard deviation which is quite high compared to standard deviation of other questionnaire statements.

Q3-Q6 are incorporated to evaluate the impact of AP-BB engagement on enhancing student understanding. AP-BB incorporates two engagement forms extracted from Naps Engagement Taxonomy (Naps, et al., 2003). These engagements are viewing and constructing. Viewing is implemented as solving visualization whereas constructing is implemented as dynamic inputs. In general, both viewing and constructing yield positive feedbacks. All statements yield an average score higher than 5 (tends to agree). Visualization (Q3-Q4) yields high impact for our respondents (average score is around 6) since most respondents are visual learners. They rely heavily on what they see when learning something. Constructing for learning Branch & Bound yields more positive feedbacks than constructing for learning Brute Force since Branch & Bound is more difficult to understand. Most respondents feel that they are required to try the algorithm multiple times with various inputs to understand how that algorithm works.

Q7-Q9 are incorporated to evaluate the impact of RCM calculator and case-based performance comparison. Q7 is incorporated to evaluate the impact of RCM calculator; Q8 is incorporated to evaluate the impact of case-based performance comparison for learning Branch & Bound characteristics; and Q9 is incorporated to evaluate the impact of displaying the number of involved instruction for learning time complexity in Brute Force and Branch & Bound. In general, all statements yield positive feedbacks since its average score is higher than 5with low standard deviation. Thus, it can be stated that most respondents tend to agree that RCM and our case-based performance comparison enhance student understanding. In addition, most respondents have similar perspective about it, since Q7-Q9 yield low standard deviation.

Q10-Q13 are incorporated to evaluate standard application aspects on AP-BB. Q10 is incorporated to evaluate application UI; Q11 is incorporated to evaluate application functionality; Q12 is incorporated to evaluate how declarative the given tutorial; and Q13 is incorporated to evaluate overall system for learning Branch & Bound strategy. Most respondents agreed that our application UI and functionality fulfill user necessity for learning Branch & Bound strategy. This finding is extracted from Q10 and Q11 average score which is higher than 6 (tends to strongly agree) with low standard deviation. Q12 yields average score lower than 6 since our tutorial contain fewer images for visual representation. Most of our respondents are visual learners who rely heavily on visual image. Thus, our textual information on tutorial is not sufficient enough for them. They require several visual images to learn the material with ease. However, our tutorial still yields positive feedbacks by respondents since its average score is still higher than 5 (tends to agree).When perceived as the whole application, AP-BB may help student to learn Branch & Bound. This finding is resulted from Q13 statement which yields the highest average score among all questionnaire statements. In addition, it also yields the lowest standard deviation which indirectly state that the most respondents have similar positive perspective about our AP-BB.

V. Conclusion and Future Works

We have proposed AP-BB, an educational tool for learning Branch &Bound strategy and its characteristics. This tool is developed based on our informal survey which indirectly states that Branch & Bound strategy is quite difficult to understand, especially for undergraduate students. This finding is also strengthened by Q2 questionnaire result where most respondents tend to agree about Branch & Bound difficulty.

In general, AP-BB incorporates two user engagements which are viewing and constructing. Both engagements yield positive feedbacks from our respondents since they agreed that these components may help students for learning Branch & Bound (Q3-Q6). In terms of its features, AP-BB consists of four modules which are Brute Force solving visualization, Branch & Bound solving visualization, Reduced Cost Matrix (RCM) calculator, and case-based performance comparison. According to our qualitative evaluation (Q3-Q9), these modules, at some extent, help student to learn Branch & Bound strategy and its characteristics. Students can learn how Branch & Bound strategy works on a particular problem and compare its impact with standard algorithm benchmark (Brute Force strategy).In addition, AP-BB has also fulfilled standard application aspects such as UI and functionality for learning Branch & Bound. This finding is based on our qualitative evaluation (Q10-Q13) where most respondents tend to agree about its fulfillment.

For further research, a controlled experiment of AP-BB will be conducted on Algorithmic Strategy course in our university. We want to evaluate the impact of this tool based on student grade improvement and in-class behavior. In addition, we also intend to incorporate more engagements from Naps Engagement Taxonomy so that our tool may enhance student understanding further.

REFERENCES

AlgoViz.org : The Algorithm Visualization Portal. (n.d.). Retrieved 12 7, 2015, from http://algoviz.org/

Areias, C., & Mendes, A. (2007). A tool to help students to develop programming skills. *The 2007 international conference on Computer systems and technologies.* Bulgaria: ACM.

Bentrad, S., & Meslati, D. (2011). Visual Programming and Program Visualization- Toward an Ideal Visual Software Engineering System -. *ACEEE International Journal on Information Technology , 1* (3), 43-49.

Buck, D., & Stucki, D. J. (2001). JKarelRobot: a case study in supporting levels of cognitive development in the computer science curriculum. *The thirty-second SIGCSE technical symposium on Computer Science Education* (pp. 16-20). Charlotte: ACM.

Carlisle, M. C., Wilson, T. A., Humphries, J. W., & Hadfield, S. M. (2005). RAPTOR: a visual programming environment for teaching algorithmic problem solving. *The 36th SIGCSE technical symposium on Computer science education* (pp. 176-180). St. Louis: ACM.

Christiawan, L., & Karnalim, O. (2016). AP-ASD1 An Indonesian Desktop-based Educational Tool for Basic Data Structures. *Jurnal Teknik Informatika dan Sistem Informasi (JuTISI) , 2* (1), 21-30.

Cisar, S. M., Pinter, R., Radosav, D., & Cisar, P. (2010). Software visualization: The educational tool to enhance student learning. *The 33rd International Convention MIPRO* (pp. 990-994). Opatija: IEEE.

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: a 3-D tool for introductory programming concepts. *Journal of Computing in Small Colleges , 15* (5), 67-71.

Debdi, O., Paredes-Velasco, M., & Velázquez-Iturbide, J. Á. (2015). GreedExCol, A CSCL tool for experimenting with greedy algorithms. *Computer Applications in Engineering Education , 23* (5), 790-804.

Elvina, & Karnalim, O. (in press). Complexitor: An Educational Tool for Learning Algorithm Time Complexity in Practical Manner. *ComTech: Computer, Mathematics and Engineering Applications , 8* (1).

Gestwicki, P., & Jayaraman, B. (2002). Interactive Visualization of Java Programs. *Symposia on Human Centric Computing Languages and Environments* (pp. 226-235). Washington: ACM.

Guo, P. J. (2013). Online python tutor: embeddable web-based program visualization for cs education. *The 44th ACM technical symposium on Computer science education* (pp. 579-584). Denver: ACM.

Halim, S. (n.d.). *VisuAlgo*. Retrieved 5 12, 2015, from http://visualgo.net/

Halim, S., Koh, Z. C., Loh, V. B., & Halim, F. (2012). Learning Algorithms with Unified and Interactive Web-Based Visualization. *Olympiads in Informatics , 6*, 53-68.

Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society. (2013). *Curriculum Guideliness for Undergraduate Degree Programs in Computer Science.* New York: ACM.

Jonathan, F. C., Karnalim, O., & Ayub, M. (2016). ExtendingThe Effectiveness of Algorithm Visualization with Performance Comparison through Evaluation-integrated Development. *Seminar Nasional Aplikasi Teknologi Informasi.* Yogyakarta: Universitas Islam Indonesia.

Learn programming with CeeBot4. (2008, 9 5). Retrieved 11 5, 2016, from http://www.ceebot.com/ceebot/4/4-e.php

Levitin, A. (2012). *Introduction to The Design and Analysis of Algorithms.* Pearson.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., et al. (2003). Exploring the role of visualization and engagement in computer science education. *ITiCSE-WGR '02 Working group reports from ITiCSE on Innovation and technology in computer science education.* New York: ACM.

Radosevic, D., Orehovacki, T., & Lovrencic, A. (2009). Verificator: Educational Tool for Learning Programming. *Informatics in Education , 8* (2).

Rajala, T., Laakso, M.-J., Kaila, E., & Salakoski, T. (2008). Effectiveness of Program Visualization : A case study with the ViLLE Tool. *Journal of Information Technology Education : Innovation in Practice , 7*, 15-32.

Shaffer, C. A., Cooper, M. L., Alon, A. J., Akbar, M., Stewart, M., Ponce, S., et al. (2010). Algorithm Visualization: The State of the Field. *ACM Transactions on Computing Education (TOCE) , 10* (3), 1-22.

Velázquez-Iturbide, J. Á., & Pérez-Carrasco, A. (2009). Active learning of greedy algorithms by means of interactive experimentation. *ITiCSE '09 Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education* (pp. 119-123). New York: ACM.

Watts, T. (2004). The SFC editor a graphical tool for algorithm development. *Journal of Computing Sciences in Colleges , 20* (2), 73-85.