GATutor: A guided discovery based tutor for designing greedy algorithm

Kavya A K Alse
IDP - ET
IIT Bombay, India
kavyaalse@gmail.com

Mukund Lahoti
Strand Life Sciences Pvt Ltd., Bangalore, India
mukundlahoti89@gmail.com

Meenakshi Verma
Google India Pvt Ltd.,
Bangalore, India
meenakshiparadise@gmail.com

Sridhar Iyer
Dept. of CSE
IIT Bombay, India
sri@iitb.ac.in

***Abstract*— Greedy algorithms is an important class of algorithms. Teaching greedy algorithms is a complex task. Ensuring that students can design greedy algorithms for new problems is also complex. We have built a guided discovery based greedy algorithm tutor (GATutor), to teach the process of designing greedy algorithms. GATutor guides the student to discover the greedy algorithm for a few well-known problems, by asking two important questions – i) what is the satisfying condition at each step? and ii) what is the selection criterion for the next item? As a result, the students not only learn the algorithms for the given problems, but also the process of designing greedy algorithms for new problems. We conducted a study to compare the greedy algorithm design abilities of the students who were trained with GATutor versus those who worked with traditional algorithm visualizations. The results indicate that students who worked with GATutor performed better in designing a greedy algorithm for a new problem. The students also said that their confidence in greedy algorithm design increased because of GATutor.**

Keywords*— *Greedy Algorithms, Guided Discovery, Teaching Algorithms, Automated Tutoring

I. INTRODUCTION

Knowledge of algorithms is critical to anyone writing computer programs. Greedy algorithms are an important class of algorithms [11]. They are useful for solving optimization problems like job scheduling and Huffman code generation [11]. Teaching greedy algorithms is a complex task [22]. Ensuring that students can design greedy algorithms for new problems is important [22].

While there are tools for teaching-learning of algorithms [17] and greedy algorithms in particular [20], they focus mostly on specific algorithms and their implementation. However, students find it difficult to transfer learning from one context to another [6]. Explicit emphasis on the underlying process is necessary for students to be able to apply the knowledge from one context to another [21]. Hence, in addition to teaching specific greedy algorithms, there is a need for explicit emphasis on the process of designing greedy algorithms, in such a way that students can apply this process to new problems.

We have designed and developed GATutor, a guided discovery based software tutor for teaching the design of greedy algorithms. GATutor guides the student to discover the greedy algorithms like Prim's, and Dijkstra's. [11].The design of GATutor involves real-life problem context, scaffolding, and immediate feedback, as recommended by the guided discovery approach [9].

Instead of simply showing the working of the algorithms, GATutor guides the student by asking two important questions – i) what is the satisfying condition at each step? and ii) What is the selection criterion for the next item? For example, in the context of Prim's algorithm, the satisfying condition is 'adding an edge should not form a cycle' and the selection criterion is 'select an edge which has least weight and connected to any already selected vertex'. GATutor guides the student to discover these selection criteria and satisfying condition on their own by asking carefully designed questions and providing visual feedback for their responses. As a result, the students not only learn the algorithms for the given problems, but also the process of designing greedy algorithms for new problems.

We conducted a 2-group quasi-experimental study to evaluate the effectiveness of GATutor. The control group learned through algorithm animations available on the web [7] and text-based study materials developed by the authors. The experimental group worked with GATutor.

The broad research goal was to see how a guided discovery based interactive tutor affects the learning of designing greedy algorithms. Specifically, we are interested in the following research questions:

1. How well are students able to design their own greedy algorithm to solve a new problem after working with GATutor, as compared to the students who worked with traditional visualizations?

2. What is the perception of students on using GATutor for learning to design greedy algorithm for a new problem?

We found that the students who worked with GATutor scored high on tasks requiring them to design greedy algorithms for a new problem. They were successful in coming up with greedy techniques, writing an algorithm and applying that in solving a problem. Students who worked with GATutor also said that it helped them to solve similar new problems.

In section 2 we analyze what work has been done towards teaching-learning of algorithms. In section 3 we explain in detail the working and features of our intervention, GATutor. In section 4 we present the research study in detail, the results of the experiments in section 5, and discussion in section 6.

978-1-4673-9509-0/15 $31.00 © 2015 IEEE
DOI 10.1109/T4E.2015.26

II. Related work

A. Teaching Algorithms

Analysis and design of Algorithms is a crucial subject in computer science. Many researchers have agreed that teaching algorithms to students is not straight-forward because each algorithm involves diverse aspects with different difficulties [5]. Traditionally, algorithms, esp. greedy algorithms are taught in the following way: present the problem; present the well-known algorithms for that problem; analyze the complexity of the algorithm and then convert the algorithm into code in a programming language [18]. Some researchers have stressed the need to teach algorithm design explicitly to students [4, 22]. They have reported that students struggle in understanding the problem and converting it to an algorithm.

Some solutions developed to address this problem include use of visuals in teaching algorithms [17, 13, 16]. There are many animations and simulations developed to teach algorithms step-by-step, with examples [17]. Some of them try to help students in translating an algorithm to a source code in a specific language [22]. Another approach to help students in learning algorithms was by providing them real life examples and solving them using an algorithm. These are specifically targeted to teach design of algorithms [10]. Along with real-life examples, the authors have argued that use of feedback in their tool has increased students' motivation.

When we consider only greedy algorithms, the traditional method of teaching is not sufficient because it directly gives the optimum selection criterion. It hides from the students, the process of designing a greedy algorithm by coming up with selection criteria themselves. Also textbooks seldom emphasize the design process which leads to the selection of optimum selection criteria [19]. GreedEx [20] is a system which teaches greedy algorithms by considering sub-optimal and optimal selection criteria. Even here, the focus is on teaching one algorithm and translating it to code. To the best of our knowledge, there is no study that tests students' ability to design greedy algorithms for a new problem.

While greedy algorithms are not universally applicable, i.e., there are classes of problems for which they don't lead to optimal solutions, they are nonetheless important. We note that students are often familiar with the use of greedy technique in daily life situations. They are used to identify some greedy-like criteria which are usually not optimal. Hence it is necessary to address these sub-optimal criteria and teach them the process of coming up with optimal criteria. One way of doing this is through algorithmic thinking.

B. Algorithmic Thinking

Algorithmic thinking is a way of defining clear unambiguous steps to reach the solution of a problem [2]. Futschek [12] has given 6 steps involved in algorithmic thinking:

1. The ability to analyze given problems
2. The ability to specify a problem precisely
3. The ability to find the basic actions that are adequate to the given problem
4. The ability to construct a correct algorithm to a given problem using the basic actions
5. The ability to think about all possible special and normal cases of a problem
6. The ability to improve the efficiency of an algorithm

We have used first three of these steps in the context of greedy algorithms to design and develop GATutor. Since we want the students to come up with their own algorithm, we have used the guided discovery learning to design GATutor. Guided discovery learning is shown to be effective for these exploratory tasks [9].

C. Guided Discovery

The students make most out of the instruction if there is an explicit demonstration of the decisions taken and actions carried out during problem solving [9]. Clark argues that guided environnment with scaffolding and real-life problems are found to motivate and help the students to connect with their earlier experiences.

Alfieri [1] after a meta-analysis of 360 studies with discovery learning, suggest guidance in the form of scaffolding, feedback, elicited explanations and worked examples. These characteristics of guided discovery were considered while designing the GATutor.

III. Implementation of GATutor

We have developed a guided discovery based tutor called GATutor, for designing greedy algorithms. It uses Job scheduling, Knapsack, Minimum Spanning Tree and Single Source Shortest distance problems as illustrative examples for designing greedy algorithms. Prims's and Kruskal's algorithm are used in Minimum Spanning Tree problem and Dijkstra's algorithm for single source shortest path.

The user can access GATutor through a browser, so it can be deployed and used in any operating system. The GATutor homepage is shown in Fig. 1.

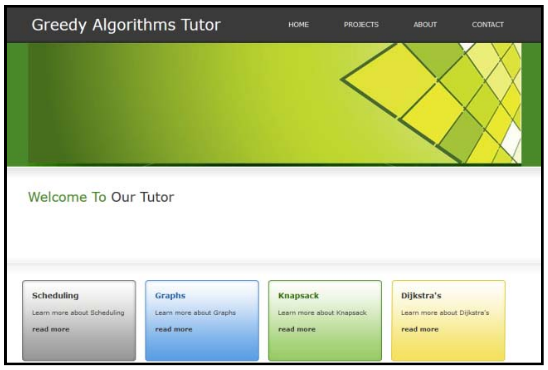


Fig. 1: Welcome Screen in GATutor

A. Objectives of GATutor

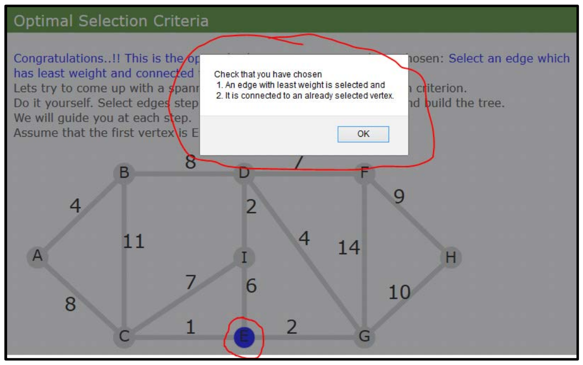
GATutor aims to explicitly teach the design of greedy algorithms and not just the implementation of specific algorithms. GATutor has the following learning objectives and the corresponding level in revised Bloom's taxonomy [3].The Students will be able to:

1. Explain what are greedy algorithms (Understand level)
2. Explain specific greedy algorithms (Understand level)
3. Analyze which selection function is optimal (Analyze level)
4. Prove its optimality by showing counter examples for non-optimal questions (Evaluate level)
5. Solve different such problems (Apply level)

B. *Design of GATutor*

GATutor has the following features based on the recommendations by guided discovery literature:

Real-life problem: Real life problems are known to motivate the students and engage them in learning. It is also helpful in getting the students to appreciate the problem and preparing them for the learning [15]. GATutor introduces a real-life problem at the beginning of each algorithm. This also serves as an indication of where the corresponding algorithm might be applicable.


Optimal Selection Criteria

Congratulations..!! This is the op hosen: Select an edge which has least weight and connected

Lets try to come up with a spanr n criterion.
Do it yourself. Select edges step nd build the tree.
We will guide you at each step.
Assume that the first vertex is E

Check that you have chosen
1. An edge with least weight is selected and
2. It is connected to an already selected vertex.

Fig. 2: Visual and explanatory feedback

Step by step building of algorithms through leading questions: GATutor doesn't provide the ready algorithm to students, but it asks well thought questions with options, and makes the students to think and come up with the algorithm themselves. In other words it follows the guided discovery method of teaching.

Interaction: By making the students to 'participate' in the algorithm design process, the GATutor ensures that important concepts are understood well by the students. This gives less chance for passive learning as in listening to a lecture or watching a video. Here the students need to actively answer the questions posed and build the algorithm according to their answers.

Visual and explanatory feedback: For the options that the students select for each question, irrespective of right or wrong, a visual and explanatory feedback is given. The feedbacks specify more than right/wrong. It gives them hints to solve the problem. For example, as shown in Fig. 2, the dialogue box says:

'*Check that you have chosen*
1. An edge with least weight and
2. It is connected to an already selected vertex.'

C. *Working of GATutor*

In general, a greedy algorithmic technique considers one candidate at a time locally, without any backtracking [11]. This selection is done by two functions: satisfying condition and selection criteria:

Finding Satisfying Condition: A satisfying condition in a greedy algorithm specifies the necessary condition for selecting the next candidate. If the satisfying condition is violated, then the resulting answer will be incorrect. The satisfying condition in Prim's is: 'Adding an edge should not form a cycle'.

Finding Optimal Selection Criteria: A selection criterion in

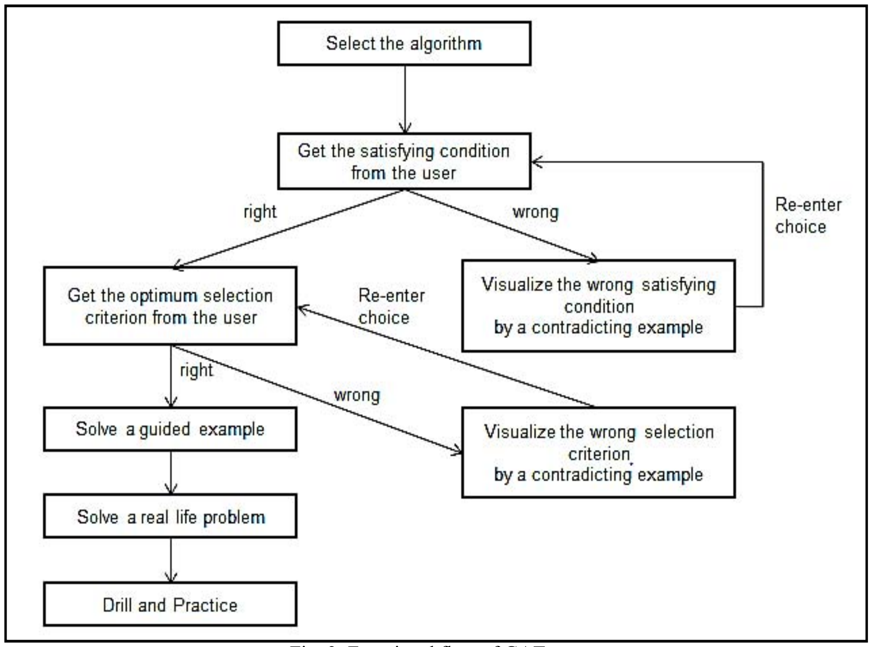

Select the algorithm
Get the satisfying condition from the user
right
wrong
Re-enter choice
Get the optimum selection criterion from the user
Re-enter choice
Visualize the wrong satisfying condition by a contradicting example
right
wrong
Solve a guided example
Visualize the wrong selection criterion by a contradicting example
Solve a real life problem
Drill and Practice

Fig. 3: Functional flow of GATutor

a greedy algorithm specifies the sufficient condition for the selecting the next candidate. If the optimal selection criterion is not used, then it will lead to a sub-optimal solution. The selection criterion in Prim's algorithm is: 'Select an edge which has least weight and connected to any already selected vertex'.

We have used these characteristics of greedy algorithms as a key feature to drive the learning in GATutor. A functional flow diagram of GATutor is given in Fig. 3. We illustrate the working of GATutor through a running example of Prim's algorithm.

GATutor starts with a real life problem and asks the student to solve the problem without any guidance. In Prim's algorithm, it gives the following real-life problem:

Government of your state wants to provide electricity to all the villages near to your city. Due to the terrain of the region the cost of laying electricity cable between two villages may not necessarily depend upon the distance between them. Given the cost data and the map of the region, Will you be able to design a topology which levies minimum cost to the state?

Then GATutor explains the problem with the required background knowledge. To teach satisfying condition, GATutor poses a question to the student and gives options of the possible satisfying conditions. These options are carefully constructed by considering students' difficulties and pre-conceptions. The question asked in Prim's algorithm is:

Can you say which among the following rules should always be followed at every step of forming a minimum spanning tree?

1. *Stop adding edges when the number of edges is n-1, where n is the number of vertices.*
2. *Adding an edge should not form a cycle*
3. *I do not know.*

When student makes a selection, a visual and explanatory response is given according to the selection. For example, if the selection is option 1, then in the visualization, n-1 edges are selected randomly and explanation is given for why the resulting graph is not a minimum spanning tree. We want the students to discover the optimal selection criteria by themselves. To make this design process explicit, we list some selection functions initially and ask the learner to discover the optimal one. The options given in the Prim's algorithm are:

Given that you have chosen the first vertex randomly from which you will start adding edges, What can be the optimum selection criteria among the following candidate functions, to select the next edge?

1. *Select an edge which has least weight and connected to a vertex you just chose*
2. *Select an edge such that the distance from first vertex to the new vertex is minimum*
3. *Select an edge which has least weight and connected to any already selected vertex*

Visual and explanatory feedback is given to the options selected in each of these questions. The students are given feedbacks depending on their choices and if their strategy is sub-optimal, then it explains why it is so and leads them to the optimal strategy. If their approach is already optimal then it explains the strategy for solving other such problems. For example, if the option selected for selection criteria of Prim's algorithm is 'Select an edge such that the distance from first vertex to the new vertex is minimum', then the students get to see the response as shown in Fig. 2. Then it guides the students to solve a problem by applying the satisfying condition and selection criterion that they have discovered.

Then it takes the students to the real-life problem introduced at the beginning and asks them to solve again, without any scaffolds.

D. Implementation

GATutor is built using Java server pages in the backend and HTML, JavaScript and SVG images in the front-end.

(i) Backend: Java servlet pages embedded with java script

We have made 87 Java servlet pages (jsp) consisting of java script code implementing variations of different algorithms to help student find out correct algorithm by themselves. We have used JSP pages for displaying the non-changing content of the system. For coding a particular algorithm, we have used java script. It receives the input by the user and accordingly changes the content of the jsp pages, then output the corresponding jsp page.

(ii) Frontend: SVG images with clickable interface

As our system is web based, using SVG images is a good option because they are scalable, and every element and every attribute in SVG files can be animated. We have used this property to make clickable images, which gives a feel like that of playing a game. As per our input these images changes their values of different elements and objects thus making the system more interactive and interesting to the user.

IV. Research study

A quasi-experimental pilot study was done to measure the efficiency of students in designing a greedy algorithm for a new problem. The study also included measuring students' perceptions about greedy algorithmic technique and GATutor. The details of the study are explained below.

A. Goal of the study

This pilot study was mainly focused to see if GATutor is successful in teaching greedy algorithmic design approach to students. The research questions considered are:

(i) How well are students able to design their own greedy algorithm to solve a new problem after working with GATutor, as compared to the students who worked with traditional visualizations?

(ii) What is the perception of students on using GATutor for learning to design greedy algorithm for a new problem?

B. Sample

Analysis and design of Algorithms course is usually taught in the second year so that they have a basic understanding of programming. Considering this, we selected 19 first year engineering students who have completed/almost completed their introductory programming course. All the students have

TABLE 1: RUBRICS FOR EVALUATING ALGORITHMS DESIGNED BY STUDENTS

LO	Correct	Partially Correct	Not Correct
LO 1	Mentions that the vertices should be considered in the increasing order of their weight. OR Mentions that the vertices with higher degree should not be considered.	Just mentions number of vertices should be maximized AND/OR Some implicit indications like growing cardinality in answer sheet	Incorrect approach
LO 2	Mentions the satisfying condition – no two vertices in the set should be directly connected. OR If the considered vertex is already connected to a vertex in the set, do not add it to the set.	Mentions but fails to recognize it as satisfying condition	Any other condition
LO 3	Mentions the optimum selection criteria – Select vertices in the increasing order of their degree. OR Leave out the vertices which have higher degree.	Mentions but fails to recognize it as selection criterion OR Any implicit notions of increasing/decreasing vertices	Any other condition
LO 4	Checks for satisfying condition at each selection, OR Checks the following condition at each selection: If the vertex is already connected to a vertex in the set, do not add it to the set.	Checks the satisfying condition/ inverted condition at some selections	Never checks
LO 5	Chooses according to optimum selection criterion at each selection. OR Discards the vertices with higher degree at each selection and the answer is correct.	Chooses according to selection criterion/ inverted selection criterion at some selections.	Never checks

either CS/Mechanical as their majors. The assignment of participant to groups was based on their scores in the introductory programming subject and a pre-test such that both the groups remain equivalent.

C. Procedure

In order to test our research questions, we chose two greedy algorithms (Minimum spanning tree and Single source shortest path) to students and then test their understanding on the following learning goals for a new problem (Maximum independent set problem). These learning goals are adapted by Futschek [12].

LO 1: The students will be able to recognize that the problem is an optimization problem. They have to precisely specify what needs to be minimized/ maximized.

LO 2: The students will be able to recognize – the satisfying condition and optimum criterion.

LO 3: The students will be able to recognize – the optimum selection criterion.

LO 4: The students will be able to write the steps according to satisfying condition involved in greedy approach for solving the given problem.

LO 5: The students will be able to write the steps according to optimum selection criterion involved in greedy approach for solving the given problem.

Since the knapsack and job scheduling problems can be directly related to daily activities, it is typical for students to have some strategy to solve those problems. Many a times such strategies turn out to be greedy even if they are not optimal [13]. So we decided to choose Minimum spanning tree (Prim's) and shortest distance (Dijkstra's) to teach the greedy algorithmic design technique and Maximum independent set problem for testing. The assumption here is, since there are less direct relationship between real life scenarios and these problems, the effect of GATutor and animations will be stronger on students. The duration of the study was around 2 hours. The primary researcher was present at the location of the study to assist in terms of logistics.

After giving instructions to students, they were asked to watch 2 videos about the basics of graphs which include the concepts of vertex, edge, degree of a vertex, paths, cycles etc. The length of the first video was 8 min and the second video, which specifically talked about spanning trees, was of 3 min duration. The students were allowed to watch them till they were confident about the content.

Then, control group students worked with the animations [5] of Prim's and Dijkstra's algorithm. Then they were given a brief lecture on satisfying condition and selection criteria in those algorithms. The experimental group students, on the other hand, worked on Prim's and Dijkstra's algorithm with GATutor. None of the students had any time restriction on these activities and on an average they took 1 hour 30 minutes before post-tests begun.

The post-test was conducted in 3 phases to test their greedy algorithmic design abilities. Phase 1: One of the questions was about the content they had just learnt – to find a minimum spanning tree. This question is used to test the understanding of the content taught by both the systems. The next problem was new to them – finding the maximum independent set using greedy technique. This involved not only problem solving, but the students were asked to design a greedy algorithm and use that algorithm to solve the problem.

Phase 2: After they have finished writing answers for these questions, they were given one additional sheet, where they were supposed to explicitly mention the satisfying condition and the selection criteria used in solving the problem. The students were free to change their answers in Phase 1 after they had attempted this phase. Our assumption was that these questions might act as probing questions.

Phase 3: Then the students were interviewed about the answers they had written. Even during the interview they were allowed to change the answer. This was followed by a survey about students' perceptions of learning and GATutor.

D. Instrument

The rubric in table 1 was developed based on algorithmic thinking [12] to evaluate the algorithm developed by the students. This rubric was used to evaluate students; answer scripts in the post-test.

TABLE 2: MEAN SCORES ON RUBRIC

	LO1	LO2	LO3	LO4	LO5
Control	1	1.2	0.56	1.67	0.67
Experimental	1.4	1.3	1	1.4	1.1

E. *Data Analysis*

The frequency and mean scores of the students according to the rubric was analyzed to test students' learning. The 5 point Likert scale data obtained from the survey was analyzed to see students' perceptions.

V. RESULTS

For the first question in the post test, which evaluated the understanding of the minimum spanning tree problem, we saw that the control group students performed better than the experimental group students. Here the optimum selection criterion was direct: In order to reduce the weight of the tree, check the weight of each edge while selecting. In the maximum independent set problem, however, the optimum selection criterion was not so obvious: The students had to find out that, in order to maximize the number of vertices in the set, the vertex with the minimum degree has to be selected at each step.

In order to answer our first research question, we measured learning using frequency and mean scores in post-test.

For answering the second question, we consider the average Likert score for each group in survey. The experimental group students felt that GATutor helped them in solving the new (maximum independent set) problem whereas some control group students disagreed that the animations helped them in solving the new (maximum independent set) problem.

A. *Learning*

The learning was measured by the frequency and mean scores obtained according to the rubric given in table 1.

Mean scores: Learning is indicated by the mean scores obtained by each group on each of the learning objectives. The mean scores are listed in the Table 2. From this table we can observe that, experimental group students have scored more marks on every learning objective but one.

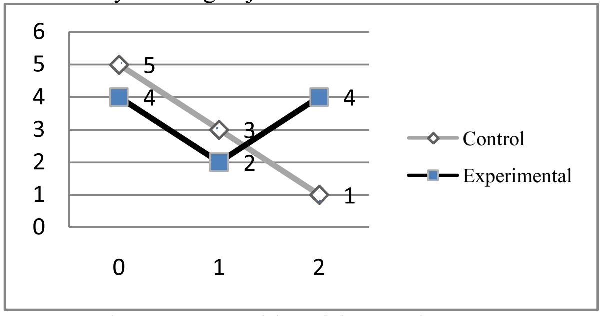


Fig. 4: Frequency of the Rubric scores for LO3

Frequency: If we zoom in, we see that the difference in mean scores in LO3 and LO5 is high. Of the five learning objectives we have, the third one is most important for greedy algorithms since it measures the concept of optimum selection criteria. Fig. 4 shows the frequency of rubric scores for both groups for LO3: The students will be able to recognize – the optimum selection criterion. The rubric scores were 0, 1 or 2 depending on whether students were correct, partially correct or not correct. Even though we couldn't test for statistical significance because of the small sample size, we observe that more students from experimental group have scored 2. On the other hand more students from control group have scored 0.

Both animation and GATutor checked for satisfying condition every time – This might be the reason behind score on LO4 being equivalent in both groups (LO4: The students will be able to write the steps according to satisfying condition involved in greedy approach for solving the given problem). But the GATutor also gave feedback on selection criteria and the animations didn't. This difference can be seen in application of selection criteria each time. Again, no statistical tests were done because of the smaller group size, but the obtained score indicate that the experimental group students score higher on an average when compared to students from control group.

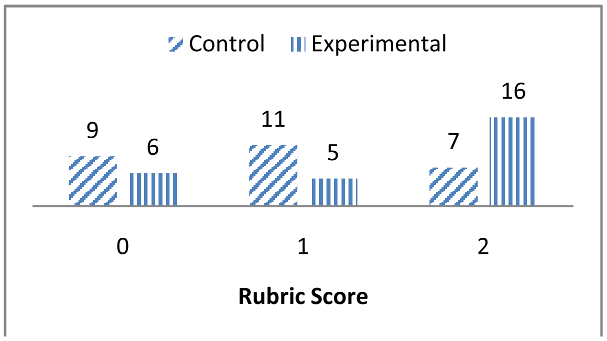


Fig. 5: Frequency of Conceptual LOs

Stratified results: We have stratified the learning objectives as conceptual and procedural based on Clark's types of content [8]. LO1, LO2 and LO3 are conceptual since they involve understanding of three properties and recognizing whether the given problem has those three properties. LO4 and LO5 are procedural because they involve applying LO2 and LO3 at each selection. Now, we consider the answer to each LO as a data point so each student will have three data points. Here also we can see that more number of students from experimental group have scored '2' in both conceptual and procedural learning objectives. This is depicted in Fig. 5.

We observe that the difference in conceptual LOs is statistically significant as found by two-tailed independent t-test [$n=27$, $t= 2.015$, $p= 0.049$] at $\alpha=0.05$.

B. *Perception*

In order to measure the attitude of students towards GATutor, we used the survey given in table 3.

The table shows that more students from the control group felt that they can design greedy algorithms for new problems.

TABLE 3: SURVEY QUESTIONS WITH RESPONSES THAT AGREE/STRONGLY AGREE

No	Statement	%Agreed (control) (N=8)	% Agreed (experimental) (N=6)
1	I have understood greedy algorithmic design technique as mentioned in the learning objective.	100	83.33
2	I found it easy to attempt the maximum independent set problem	75	100
3	I will be able to design a greedy algorithms for similar* problems. (*minimum spanning tree & Maximum independent tree type of optimization problems)	62.5	66.67
4	GAtutor helped me understand the concepts related to greedy algorithmic design, as mentioned in the learning objectives.	87.5	83.33
5	GATutor helped me in solving the maximum independent set problem.	62.5	83.33
6	GATutor helped me learn greedy algorithm design for similar problems.	87.5	50

*(For control group, GATutor in the survey was replaced by animation and study material)

But less number of students from the experimental group felt the same. But for solving the maximum independent set problem, more students from experimental group felt confident.

VI. Discussion and Conclusion

The first of the two research questions we considered was, how good is GATutor, for students to come up with their own greedy approach to solve a problem? In the experimental group, those who showed evidence of LO2 and LO3 (Conceptual understanding of satisfying condition and selection criteria) in phase 1, were also able to explicitly state LO2 and LO3 in the phase 2. This shows that the selection criteria they used were not intuitive but conscious decisions. This is also supported by the survey results. Most of the control group students on the other hand, did not show evidence and were not able to state it properly even when they found the optimum selection criterion.

Another observation we made was most of the students who worked with GATutor had selected total/partial greedy optimum selection criterion. The reason might be, GATutor taught them recognizing and applying the satisfying condition and selection criteria and not directly solving the problem. It was the students who selected the options, 'saw' whether it was correct or not and decided themselves. So it was easier for them to choose the 2 conditions while designing a greedy algorithm for a new problem.

In both LO3 and LO5, the experimental group has scored almost double the control group. Here, the experimental group students have 'seen' different selection criteria, and got explanatory feedback at every selection. In maximum independent set problem, this directed students to try many selection criteria while selecting vertices. These are supported by the following anecdotes from the interview.

"…then I tried the outermost ones (vertices) and I found the way to find the maximum cardinality… It has the least number of edges connected to it…"

"…All these have too many interconnections. So instead of selecting these we can select (vertices) A and K (with least degree)…"

"…An animation can only help you learn its content. Any doubt, clarification, or further application, may not be easily answered…"

The second research question was about the perception of the students about GATutor and whether they thought GATutor/animation helped them in solving the maximum independent set problem. It is interesting to see that all the students in the experimental group found it easy to attempt the maximum independent set problem even though only 50% of them felt that GATutor helped them to learn designing of greedy algorithm. On the other hand only 75% of the students in control found it easy to solve maximum independent set problem and 87.5% of the students think that animations and the study material helped them to learn greedy algorithm design for similar problems. But as we observe from the post-test, experimental group students were better in designing greedy algorithms than control group students.

These results suggest that animations are better for solving applying the same algorithm in different situations. A guided discovery based tutor that leads the students through design decisions is better for them to learn designing algorithms. GATutor can become an interesting tool to teach design of greedy algorithms along with classroom teaching. But since this is a pilot study with less number of participants the generalizability of the results might be less.

Acknowledgment

We would like to acknowledge Prof. Sahana Murthy, IDP – Educational Technology, IIT – Bombay for guidance in research design. We are thankful to Chakrapani D.S, Dept. of CSE, JNNCE, Shimoga and Alaka Ananth, Dept. of CSE, NIE, Mysore.

References

[1] Alfieri, L., Brooks, P. J., Aldrich, N. J., & Tenenbaum, H. R. Does discovery-based instruction enhance learning? *Journal of Educational Psychology, 103* (1), Feb 2011, pp. 1 - 18.

[2] *Algorithmic Thinking*. Retrieved July 19, 2015, from Teaching London Computing: http://teachinglondoncomputing.org/

[3] Anderson, L. W. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives.* Allyn & Bacon, 2001.

[4] Baeza-¥ates, R. A. Teaching algorithms. *ACM SIGSAT News*, 1995, pp. 51-59.

[5] Begosso, L. C., Begosso, L. R., Gonçalves, E. M., & Gonçalves, J. R. An approach for teaching algorithms and computer programming using Greenfoot and Python. IEEE *Frontiers in Education Conference,* 2012, pp. 1-6

[6] Bransford, J. D. & Schwartz, D. L. Rethinking transfer: A simple proposal with multiple implications. *Review of research in education, 24*, 1999, pp. 61-100.

[7] Chinneck, J. W. Retrieved June 29, 2015, from Practical Optimization: A Gentle Introduction: http://www.sce.carleton.ca/faculty/chinneck/po.html, 2006.

[8] Clark, R. C., & Mayer, R. E. *E-Learning and the Science of Instruction: Proven guidelines for consumers and designers of multimedia learning.* San Francisco: Pfeiffer, 2007

[9] Clark, R. E. How much and what type of guidance is optimal for learning from instruction. In T. M. Sigmund Tobias, *Constructivist Instruction: Success or Failure?* 2009, pp. 158-183

[10] Combéfis, S., & de Saint-Marcq, V. L. C. Teaching Programming and Algorithm Design with Pythia, a Web-Based Learning Platform. *Olympiads in Informatics*, 2012, pp. 31-43.

[11] Cormen, T. H. *Advanced Algorithms-CS 6/76101.* The MIT Press, 2001.

[12] Futschek, G. Algorithmic thinking: the key for understanding computer science. *Informatics education–the bridge between using and understanding computers. Springer Berlin Heidelberg*, 2006, pp. 159-168.

[13] Ginat, D. The greedy trap and learning from mistakes. *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education SIGCSE'03,* 2003

[14] Naps, T. L. JHAVÉ: Supporting Algorithm Visualization. *Computer Graphics and Applications*, 2005, pp. 49-55.

[15] Roll, I., Holmes, N. G., Day, J., & Bonn, D. Evaluating metacognitive scaffolding in guided invention activities. *Instructional science*, 2012, pp. 691-710.

[16] Shaffer, C. A. Algorithm visualization: The state of the field. *ACM Transactions on Computing Education*, 2010.

[17] Ullrich, T., & Fellner, D. AlgoViz-a computer graphics algorithm visualization toolkit. *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, 2004, pp. pp. 941-948.

[18] Velázquez-Iturbide, J. Á. The design and coding of greedy algorithms revisited. ACM *Innovation and technology in computer science education,* 2011, pp. pp. 8-12

[19] Velázquez-Iturbide, J. Á. An Experimental Method for the Active Learning of Greedy Algorithm. *Transactions on Computing Education, 2013*

[20] Velazquez-Iturbide, J. A. GreedEx: A Visualization Tool for Experimentation and Discovery Learning of Greedy Algorithms. *Transactions on Learning Technologies, 6*(2), 2013, pp. 130-143.

[21] Woods, D. R. Helping your students gain the most from PBL. *Asia-Pacific Conference on PBL.* Singapore, 2000

[22] Yoo, J. Y. Can we teach algorithm development skills? ACM *Proceedings of the 50th Annual Southeast Regional Conference*, 2012, pp. 101-105