Identification and Removal of Misconceptions on Optimization Concepts Underlying Greedy Algorithms

J. Ángel Velázquez-Iturbide
Departamento de Lenguajes y Sistemas Informáticos I
Universidad Rey Juan Carlos, Móstoles, Madrid, Spain
Email: angel.velazquez@urjc.es

We have conceived a didactic method for the active learning of greedy algorithms which has been used for five years in an algorithms course. As a part of this experience, we analyzed the reports elaborated by students in three consecutive academic years. In this article we present our findings, which include several unexpected misunderstandings of students. We also describe the interventions addressed, which resulted in the removal of the most severe misunderstanding. The main contributions of the article are two-fold. On the one hand, we identify several misunderstandings on basic optimization concepts that underlie greedy algorithms. On the other hand, we present the successful removal of misconceptions thanks to an adequate combination of: didactic method, educational software, scheduling of classes and lab sessions, and teaching materials.

ACM Classifications: *F.2.2 (Analysis of Algorithms and Problem Complexity): Nonnumerical Algorithms and Problems – sequencing and scheduling; K.3.1 (Computers and Education): Computer Uses in Education; K.3.2 (Computers and Education): Computer and Information Science Education – computer science education.*

Keywords: *Greedy algorithms, GreedEx system, misconceptions, grounded theory, didactic intervention*

1. Introduction

Programming education has been a challenge for computer science instructors for over forty years (Robin *et al*, 2003). Therefore, it is not surprising that computer scientists, pedagogues and psychologists have devoted great efforts to enhance its teaching.

An outstanding area of research is the understanding of students' difficulties (Fincher and Petre, 2004). The subjects of this research are students' conceptual and mental models, their perceptions and misunderstandings. The models under study range from general (e.g. learning differences between novices and experts) to specific (e.g. misunderstandings on recursion).

We must distinguish between conceptual and mental models (Norman, 1983). A conceptual model is a representation of knowledge, built by the instructor in order to transmit it to the students. Consequently, a conceptual model must be precise, complete and consistent. On the other hand, a mental model is the presentation that a student builds of a conceptual model, i.e. his/her understanding of it. Consequently, a mental model often is partial and ambiguous.

We do not speak about correct mental models because they vary from one person to another. It is a well-known fact that two students receiving the same instruction may construct different mental

Manuscript received: 19 January 2013
Communicating Editor: Francisco José García Peñalvo

models (Ben-Ari, 2001). Therefore, we speak about viable or unviable models, depending on whether they allow explaining and understanding the phenomena under observation. We may speak about conceptual errors or misconceptions to refer to key aspects of a mental model that make its viability difficult or impossible. In addition, the term preconception is used to refer to ideas which are previous to instruction in a subject. From the point of view of constructivism (Ben-Ari, 2001), knowledge is constructed from previous knowledge and instruction. Therefore, pre- and misconceptions can be used by the instructor to assist students in a more successful construction of their knowledge.

In this article, we present a research conducted for three consecutive academic years about misunderstandings on greedy algorithms. We may highlight two main contributions. On the one hand, we identify misunderstandings on basic optimization concepts that underlie greedy algorithms. On the other hand, we exemplify a case of action research (Clear, 2004) conducted to enhance the teaching of greedy algorithms. The identification of misunderstandings and our subsequent interventions resulted in a substantial reduction of them and in the removal of the most severe ones.

The structure of the article is as follows. Section 2 describes related work on misconceptions in algorithms education. In Section 3 we present a brief summary of our didactic method. Section 4 describes in detail the protocol and results of the evaluation conducted in the first academic year. Section 5 describes the changes in protocol in the two subsequent evaluations, the interventions performed and the evolution of results. Finally, Section 6 contains a discussion and future work.

2. Related Work

A number of studies have been conducted in computer science education on students' difficulties since the seventies, mainly on learning to program. These studies address different programming paradigms, from procedural or object-oriented programming to functional, logic or concurrent programming. Please refer to the survey by Clancy (2004) for an introduction. A consequence of these studies is that we currently know the marked behavioural differences between novice and expert programmers (Winslow, 1996). Studies have also been conducted regarding the problem-solving process of students; e.g. read the review by McCauley *et al* (2008) about debugging.

There are fewer studies on preconceptions or misconceptions on algorithms. Haberman *et al* (2005) noticed the fragile knowledge of students on algorithms. Students hardly distinguish different elements of the algorithm development process, such as the algorithm itself, its underlying design decisions and analyses results.

Some studies have been aimed at discovering students' misconceptions on properties of algorithms. McCartney *et al* (2009) describe alternative conceptions of the definition of algorithm efficiency, such as worst-case analysis, average-case analysis, better for more cases, better in all cases, better for real-world scenarios, and others. Gal-Ezer and Zur (2004) inquired whether some intuitive rules, which are at the origin of many science and mathematics misconceptions, also affect the concept of algorithm efficiency. They found that this seemed to be true for some rules, so students tended to consider that shorter algorithms or algorithms with fewer variables were more efficient. Kolinkant (2005) studied misconceptions regarding correctness. She found that students are more tolerant to bugs than expert programmers, therefore they share a weaker definition of correctness.

Regarding algorithm design techniques, Danielsiek *et al* (2012) have reported on the difficulties of students in mastering dynamic programming, and some misconceptions of the divide-and-conquer technique. The only misunderstanding on greedy algorithms we have found in the

literature does not refer to greedy algorithms themselves but to the improper and hasty use of the greedy technique to solve combinatorial problems (Ginat, 2003).

Some studies have been conducted regarding specific algorithms. Simon *et al* (2006) studied students' preconceptions on sorting. Most of their findings were about programming skills but they also found preconceptions on how to compare numbers as strings of digits. More recently, Taherkhani *et al* (2012) analyzed students' implementations of sorting algorithms. Both studies identified different approaches to sorting. Many solutions embedded the most popular sorting algorithms (i.e. selection sort, insertion sort, bubble sort, merge sort and quick sort, among others) but others included original although extremely inefficient algorithms. Taherkhani *et al* (2012) also found inefficient variations of direct sorting algorithms. Heaps and their manipulation algorithms have been reported to be problematic, too (Danielsiek *et al*, 2012; Sepälä *et al*, 2006).

Typically, there are two phases in the methodology for addressing misconceptions (Clancy, 2004): detecting and identifying them, and helping students in enhancing their mental models. There are experiences of successful interventions to remove computer science misconceptions. For instance, Sanders *et al* (2006) report on their interventions, with positive results, to remove severe mis–conceptions on recursion (Götschi *et al*, 2003).

There are different approaches to interventions. One approach consists in generating or presenting cognitive conflicts to students, in the hope that students will re-elaborate their mental models so that conflicts will disappear. This approach has been successfully put into practice in the software animation field, either based on known misconceptions (Ma *et al*, 2011) or by automatically generating the misconceptions (Moreno *et al*, 2010). A different approach gives more emphasis to knowledge integration, where ideas are more distinguished, structured and connected (Linn, 1995).

3. Experimental, Didactic Method for Greedy Algorithms

Several years ago we proposed a didactic method for the active learning of the foundations of greedy algorithms (Velázquez-Iturbide and Pérez-Carrasco, 2009). The method is intended to assist students in learning the role of selection functions in greedy algorithms. In brief, the student is given an optimization problem to be solved by a greedy algorithm and a collection of candidate selection functions, and he/she has to decide which of them are optimal. The method was described in full detail elsewhere (Velázquez-Iturbide, 2013; Velázquez-Iturbide and Debdi, 2011), therefore we simply sketch it here.

For instance, consider the activity selection problem (Cormen *et al*, 2009; Kleinberg and Tardos, 2006). Given a set of n activities, each one with a start time s_i and a finish time f_i, we must build a maximum-size subset of non-overlapping activities. We may consider six "reasonable" selection functions for the activity selection problem: increasing or decreasing order of start time (denoted S↑ and S↓, respectively), increasing or decreasing order of finish time (denoted F↑ and F↓, respectively), and increasing or decreasing order of duration (denoted D↑ and D↓, respectively). In addition, other selection functions can be stated to make more complex the experimental process, e.g. increasing or decreasing order of activity index (denoted I↑ and I↓, respectively).

The inquiry process is simple and proceeds according to the scientific method. Assume that the problem to solve targets maximization (as is the case of this problem). The student must create some input data and determine the result of the greedy algorithm resulting from applying each of the selection functions to these input data. Those selection functions that yield lower values can be discarded. The student will repeat this process with new input data until he/she may hypothesize which selection functions are optimal, i.e. they always yield optimal values.

For this problem, the student should induce that two of the selection functions are optimal (S↓ and F↑). There is a probable risk that he/she also proposes D↑ to be optimal, given that this selection function gives an optimal result in over 95% of the cases (Velázquez-Iturbide and Debdi, 2011).

The method is simple and can be conducted mechanically, for instance by generating random input data. Although it is natural to experiment randomly in the initial stages, so that one gets familiarized with the problem and the system, it is later preferable to spend some time designing input data that show that some selection function is not optimal. In other words, it is more convenient to spend some time designing counterexamples of the optimality of such a selection function. This claim proves to be particularly valuable to find a counterexample for nearly-optimal selection functions such as D↑.

The experimental method can be conducted manually, but it is more efficient and enjoyable to have the assistance of an interactive system. We have built a system, called GreedEx (Velázquez-Iturbide *et al*, 2013), that supports the experimental method for six optimization problems, namely the activity selection problem (Cormen *et al*, 2009; Kleinberg and Tardos, 2006) and five knapsack problems: the fractional and the 0/1 knapsack problems (Brassard and Bratley, 1996; Cormen *et al*, 2009), as well as three variations with different maximization measures (number of objects introduced into the knapsack, weight of these objects, and number of objects introduced into two knapsacks). Notice that the two latter problems and the 0/1 knapsack problem cannot be solved optimally with greedy algorithms.

We have used our didactic method for five consecutive academic years in a course on design and analysis of algorithms offered to junior students enrolled in a computer science major (Velázquez-Iturbide, 2012; Velázquez-Iturbide, 2013). In each academic year, we made use of one or several lab sessions for a learning-by-discovery assignment based on our experimental method, which also were used to evaluate GreedEx usability. In three consecutive evaluations we analyzed the reports elaborated by students to assess their performance. In the fourth section of this article we describe the first evaluation in detail and in the fifth section, the second and third evaluations. More details about the three evaluations can be found in a technical report (Velázquez-Iturbide, 2011).

4. First Evaluation

In this section we present the procedure used for the first evaluation, as well as the results obtained. This evaluation was held in January 2009 (academic year 2008/2009).

4.1 Assignment Sessions

The instructor introduced greedy algorithms in two sessions at the classroom, also using the GreedEx system in the second session.

In the third session, two hours long, students had to solve an assignment in a computing lab. At the beginning of the session, the students downloaded all the materials necessary for the assignment: the assignment statement, the GreedEx system[1], and a report template. In addition, they were given an opinion questionnaire in paper (to evaluate GreedEx usability). The assignment statement contained the problem statement and a brief description of GreedEx. The students had to perform several tasks, either individually or in pairs:

1 Actually, an interactive system, called SEDA (Velázquez-Iturbide and Pérez-Carrasco, 2009) was used in both lab sessions. It was an antecessor of GreedEx, with the same features but restricted to the activity selection problem. For simplicity, we refer to GreedEx in this evaluation.

- Use GreedEx to identify the optimal selection functions for the activity selection problem. The selection activity problem was chosen because of the difficulty of discovering the non-optimality of the D↑ selection function.
- Fill and submit electronically a short report, written using a simple Word template. Students had to identify the optimal selection functions, give an informal rationale and provide evidence of their optimality, both summarized and in detail.
- Fill and deliver the opinion questionnaire (voluntary).

We gathered 18 reports corresponding to 13 pairs of students and 5 individuals. For the presentation of results in the following sections, we will denote by G1 the report of the first group, by G2 the report of the second group and so on.

After this session, the instructor gave two additional theory classes, where key experimentation concepts were revisited.

Finally, a second lab session was held. The students had to do this assignment organized in the same groups as in the previous lab session. The assignment statement identified the two optimal selection functions (i.e. S↓ and F↑), and asked students to review their previous report and to explain the mistakes committed (both proposals of suboptimal selection functions and lack of proposal of optimal selection functions). They had to document their results in another report model. We gathered 11 reports, corresponding to 9 pairs and 2 individuals.

4.2 Analysis of Reports

A first analysis of the reports of both sessions was made in March 2009. For each report, we annotated: the selection functions proposed, the explanation given on the optimality of each selection function, the experimental evidences given for each selection function, and the global rationale. We also recorded data about GreedEx usage: number of data sets documented and described, size of such data, and number of selection functions executed for each data set. Following the principles of grounded theory (Glaser and Strauss, 1967), we started with very few predefined categories. We only foresaw two factors, namely correctness of the solution and consistency in the experimental reasoning.

Given the poor writing skills of students, we had to read each report several times until we considered that we had fully understood their proposals. Finally, we elaborated a provisional classification into four categories (which roughly align with groups of categories A-B, C, D-E and F-G in the next section).

In January 2011 we again analyzed the reports of this first evaluation, reading each report several times. Some of the elements recorded for each group were refined. Factor 2 became more important thus, instead of exclusive categories, we preferred to differentiate factors 2, 3 and 4 described in the next section.

In May 2011 we analyzed the reports of the third and second evaluations (in this order). Factor 1 became more important, making a last revision of the reports of this first evaluation.

4.3 Results of the First Session – Factors and Categories

We may differentiate four main factors that characterize wrong reports:

1. **Suboptimal selection functions**. Discarding a suboptimal selection function depends on having found a data set which is a counterexample of its optimality. For the selection function D↑,

the task is harder than for other selection functions because it is nearly optimal: although in the worst case it may select half the activities compared to the optimal selection functions (Kleinberg and Tardos, 2006), we have checked experimentally that it is optimal in over 95% of the cases (Velázquez-Iturbide and Debdi, 2011). Only 4 of the 18 groups found a counterexample for the selection function D↑.

2. **Inconsistent reasoning.** The results obtained from each data set must guide students' provisional conclusions about the optimality of the different selection functions, and even guide the design of new input data.

   However, some groups were inconsistent in different ways: they based their proposal on one single input data set while ignoring the results of other data sets (G2); they discarded selection functions without having found a counterexample (G8); or they made different proposals in different parts of their report (G5, G11, G12, G13, G16).

3. **Additional optimization criterion.** In general, an optimization problem may have several optimal solutions for given input data. Furthermore, it can be solved optimally by several selection functions. For instance, the minimum-cost spanning tree problem (Brassard and Bratley, 1996; Cormen *et al*, 2009; Kleinberg and Tardos, 2006) can be solved with the Prim's and Kruskal's algorithms, among others.

   However, a number of students believed that only one optimal selection function was possible. Consequently, several groups restated the target function by incorporating an additional optimization criterion referred to the total amount of time of the selected activities. Four groups proposed maximizing it (e.g. occupation time of a room) while six groups proposed minimizing it (e.g. waiting time to complete an activity). In this way students obtained the "most optimal solution".

   For instance, let us read the rationale given by G5: "But, in these strategies, our reference to choose «the best of the best» was that strategy with the longest duration, because it makes a better use of time. (…) Therefore, according to this criterion, for these input data, the best strategy is increasing order of index [0,3,4,10], since it performs four activities and its time length is 11 units of time."

4. **Dependence on input data.** An optimal selection function must give the best solution for any input data. Suboptimal selection functions may give an optimal value for some input data but they give a suboptimal value for other input data.

   However, some groups thought that the optimal selection function could vary, depending on the specific input data. Interestingly, this factor is always present with the previous one, but not the other way.

   We cite G5 again: "On solving the problem, we have found five selection functions that produce an optimal result, given that they perform four activities. The results obtained exclusively depend on input data since, if they change, the optimal selection functions could vary." Here, they proposed five selection functions. In a posterior section, they claim: "For these input data, the best selection functions are those that perform six activities." Here, they only proposed three selection functions.

The solution proposed by each group exhibits a subset of the four factors. In principle, there are 2^4=16 different combinations, but the factors are not independent. In effect, no group randomly guessed the optimal selection functions, thus students who did not exhibit factor 1 (i.e. who identified the optimal selection functions), neither exhibited the other factors. Consequently,

seven combinations were never present. In the same way, we only found factor 4 jointly with factor 3, therefore the combinations 1-4 and 1-2-4 did not occur. In summary, there are seven categories of answers left, as Table 1 shows.

Category	Factors	Groups
A	–	G1, G3, G4, G17
B	1	G6
C	1-2	G2, G8, G13
D	1-3	G7, G15
E	1-2-3	G11, G12, G16
F	1-3-4	G9, G10, G14, G18
G	1-2-3-4	G5

Table 1: Clasification of Answers into Categories

The seven categories are very different with respect to the viability of their mental models. We may group them into three degrees of viability, which are shown in decreasing order of viability.

The least severe misconception corresponds to factor 1. For the activity selection problem, it is easy to propose the suboptimal selection function D↑, even when the underlying concepts are well understood. Therefore, categories A and B are viable, given that the problem is solved consistently and (almost) correctly.

Next, categories C and D contain one additional misconception, either on the experimental method or on the additional optimization criterion. In these cases, we find severe conceptual problems, but they are limited.

Finally, categories E, F and G at least exhibit two additional misconceptions. We consider that their mental models are unviable.

4.4 Results of the First Session – Supporting Input Data

It is interesting to notice the great influence of the problem statement and of the default settings of the GreedEx system on the students' choice of input data. All the groups but two (i.e. 16 groups) tested the example included in the problem statement, with size 11. In addition, 11 groups included some additional example of size 11. The second most frequent size was 12 (5 times), which is the default size of data generated by GreedEd.

Most groups were not very proficient in providing experimental evidence. However, the quality of the evidence provided by groups in categories A and B was usually higher than that provided by other groups. This fact becomes apparent in several ways. Firstly, the only group (G1) that provides two manually designed counterexamples of D↑ belongs to category A.

Secondly, groups from categories A and B usually provide more structured evidence by selecting a subset of all the executions they performed. The first column of Table 2 shows the number of examples selected by each group, as well as the total number of executions performed (as far as we may deduce from their reports). They are sorted in decreasing order of number of executions (the higher number of executions, the more evidence obtained) and then in increasing number of

examples provided (the fewer number of examples provided, the more elaborated selection). The second column identifies the groups that documented a specific number of examples and executions. The third column contains the categories of these groups.

No. examples/no. executions	Groups	Categories
2/13	G3	A
2/11	G17	A
2/9	G8	C
2/5	G6	B
5/5	G1	A
4/4	G2, G4	A, C
3/3	G15	D
2/2	G5, G7, G9, G10, G11, G12, G13, G14, G16, G18	C, D, E, F, G

Table 2: Number of examples and executions shown as evidence

Notice that groups G3 and G17, of category A, respectively performed 13 and 11 executions but only provided evidence with two examples. They were persistent because they did not find a counterexample of D↑ before input data 10 and 9, respectively. At the other extreme, ten groups (none of them belonging to categories A or B) only provided evidence of having tried with two examples.

Note also that three of the four groups who provided a subset of the total of executions as evidence were consistent in their reasoning (the exception was group G8).

Thirdly, the five groups who provided evidence by means of the historic table (G1, G3, G6, G8, G17) belonged to categories A-C. (The historic table is a table of GreedEx that displays a summary of the session, showing the result of each execution for every input data and selection function.). From the historic table, we also know that two groups from categories A and B (G3, G6) only provide two detailed examples, but they were selected from the 13 and 5 executions that they document, respectively.

4.5 Results of the First Session – Justification

The groups justified their proposal in very different ways. For their classification, we use the categories identified by McCartney *et al* (2009). In our own words:

- **Convincing.** The group gave an informal but logical reasoning about why a particular order of selection yielded a higher number of activities.

  For instance, group G1 justified the F↑ selection function as follows: "The rationale of this strategy is very simple: by using increasing order of finish time, we make sure that we first introduce those activities which sooner leave the sports facilities free. So, more sport combinations can occur without overlapping."

  A remarkable fact is that three groups (G3, G4, G6) gave arguments based on the symmetry of selection functions S↓ and F↑. They first justified one of these selection functions and then justified the other one with a symmetry argument.

Thus, G3 gave the following explanation: "The second one (Increasing Order of Start Time), picks the activities up by their start time. (…) It is similar to the previous strategy, but reversed. (…)"

- **Partial.** The argument is reasonable but not as precise as in the previous category or it only considers a subset of the issues which are important for optimality.

  For instance, group G8 justified the suboptimal selection function D↑ as follows: "On considering the activities with the shortest duration, it is more probable that we have the highest number of activities that can be performed in a given lapse of time. We may remark that the strategies that take into consideration start or end time always are going to depend on the case, since the most important issue to perform more activities in a period of time is activity duration." Independently of the correctness of their argument, notice that this group argues that duration is the most relevant factor for optimality but they do not address the overlapping issue.

- **No support.** These groups gave an argument that does not support their proposal. This happened in several ways: some groups described the selection function they proposed, others explained the result of the proposed selection functions by simulating their application, and others simply are confusing.

  The "descriptive" justification is the most common in this category (3 out of 6 groups). Let us see the justification given by G13 for F↑: "The activities that finish earlier have a preference. If there is a draw, the one starting earlier. If still there is a draw, the one firstly introduced in the program." Its justification of other selection functions is analogous.

  An example of a confusing justification follows, given by group G2. Actually, it seems to be justifying D↑ with arguments adequate for D↓: "Taking logics into consideration, in order to choose the highest number of non-overlapping activities, the tactic would be to select those activities that we can perform given their duration, so we will choose those with long duration and that do not overlap, so we select those activities that demand the longest time without overlapping with any other activity that we may perform at the same time. The strategy that better adapts to this criterion is increasing order of duration. The others with cardinality 4 are equally valid, but they are based on criteria such as start time or the index it occupies in the table, which are actually used mathematically."

- **No argument made.** Finally, some groups did not include any rationale for their proposal, they restated the target optimization function or they identified again the proposed selection functions.

Table 3 contains the list of argument categories, as well as the groups that used these arguments and their corresponding classification categories. Notice that groups with convincing arguments only correspond to categories A and B. Groups who only provide partially convincing or no supporting arguments correspond to the rest of categories. Finally, all the groups who do not give any justification correspond to categories E-F.

Category of justification	Groups	Categories
Convincing	G1, G3, G4, G6, G17	A, B
Partial	G7, G8	C, D
No support	G2, G5, G10, G13, G15, G16	C, D, E, F, G
No argument	G9, G11, G12, G14, G18	E, F

Table 3: Categories of justification

A minor observation is that five groups (G7, G10, G14, G15, G18) explained their choice using real-world situations: persons, rooms or processors. All of these groups share factor 3 (an additional optimization criterion) and no group gave a convincing justification.

4.6 Results of the Second Session

Let us remind that the report of the second session was delivered by 11 groups out of the 18 groups who participated in the first session. The seven absent groups are divided into five who had their assignment incorrect (G2, G7, G12, G15, G16) and two correct (G3, G4).

Two groups who had performed their assignment well (G1, G17) confirmed it and, in one case (G1), a mundane erratum was fixed. From the remaining nine groups, seven groups fixed their assignment (G5, G6, G8, G9, G10, G13, G14) and two did not (G11, G18), apparently without any relation to their previous categories.

Consequently, this second session was useful for half the groups with the assignment of the first session incorrect (7 out of 14) since they were successful in fixing it.

5. Second and Third Evaluations

The second and third evaluations were held in November 2009 (academic year 2009/2010) and November 2010 (academic year 2010/2011). In these academic years, several didactic interventions were made in a trial to remove the misunderstandings detected.

5.1 Didactic Interventions and Changes in Protocol

We may remark the following changes in the second evaluation:

- **Classroom.** Experimentation concepts were introduced in the classroom session previous to the first assignment session. Students also were exposed in between the two lab sessions to problems non-solvable by greedy algorithms, e.g. the 0/1 knapsack problem (Brassard and Bratley, 1996; Cormen *et al*, 2009).
- **Assignments.** The assignment statement and the report template were slightly modified. In the statement, students were required to identify "one or several" optimal selection functions (previously, we required "the" optimal selection function). In the report template, the possibility of proposing several optimal functions was made more explicit. Students also were warned about the fact that both sessions were parts of the same assignment, in order to encourage them to attend to the second session.

In the third evaluation, we introduced more drastic changes:

- **Assignments.** The second session of the chapter was exclusively devoted to familiarize students with the didactic method, warning them about its preparatory character for the actual assignment. It consisted of one hour in the classroom, devoted to experimentation concepts, and one hour in the computer lab, where students exercised with the GreedEx system and the knapsack problem. This problem was chosen because it is very easy to find out which its optimal selection functions are. The report template was again slightly modified.
- **Teaching materials.** We elaborated lecture notes that explicitly addressed the misunderstandings detected. They consisted of eight pages, structured into four sections: optimization problems, the greedy technique, the experimental method, and the GreedEx system.

The number of reports gathered in the second evaluation was 21 in the first session and 13 in the second one. In the third evaluation, we gathered 23 reports in the first session and 11 in the second one.

In the second evaluation, students used the same version of the interactive system as in the first one, and in the third evaluation students used GreedEx. Let us remind that GreedEx supports six problems, including the two problems used in the three lab sessions.

We must remark an important enhancement implemented in GreedEx. An "intensive execution" function was incorporated, consisting in the possibility of executing all the selection functions over a high number of randomly generated input data sets. This facility eases discovering the sub-optimality of D↑ for the activity selection problem.

5.2 Evolution of the Results

The reports gathered share features with the reports of the first evaluation, without additional factors or categories. More details about these sessions can be found elsewhere (Velázquez-Iturbide, 2011). The evolution of categories in the three evaluations is shown in Table 4.

Categories	Factors	1st eval.	2nd eval.	3rd eval.
A	–	22.22%	28.57%	47.83%
B	1	5.56%	23.81%	39.13%
C	1-2	16.67%	23.81%	13.04%
D	1-3	11.11%	19.05%	0%
E	1-2-3	16.67%	4.76%	0%
F	1-3-4	22.22%	0%	0%
G	1-2-3-4	5.56%	0%	0%

Table 4: Evolution of Percentages of Categories

Notice that there is an improvement in the results of the three evaluations. In the second evaluation, categories F-G disappear (specific to factor 4, i.e. dependence on input data), while in the third evaluation categories D-E disappear (specific to factor 3, i.e. additional optimization criterion). We also notice an increase in the percentage of students who find the correct solution (up to 47.83%).

The improvement can be better appreciated if we group the categories by viability, as Table 5 shows. Notice that students with an (almost) correct solution (categories A-B) grow up to 86.96% in the third evaluation.

Groups of categories	1st eval.	2nd eval.	3rd eval.
A-B	27.78%	52.38%	86.96%
C-D	27.78%	52.86%	13.04%
E-F-G	44.45%	4.76%	0%

Table 5: Evolution of Percentages of Groups of Categories

6. Discussion

6.1 Identification of Misconceptions

We were very surprised with the results of the first evaluation, after detecting a high number of groups with severe and unexpected misunderstandings. Had we not used the GreedEx system to

support the experimental assignment, we would hardly have detected the misunderstandings. Probably, the assignment would have been more traditional, consisting in the design and coding of a greedy algorithm, thus the difficulties would have been different, e.g. buggy algorithms.

The misunderstandings identified are central to the definition of the optimality property. Many students seem to share a weaker model of optimality than experts: they do not force a given algorithm to be optimal for all its valid inputs, but they accept having different optimal algorithms for different input data. This tolerance is consistent with findings about other algorithm properties: correctness and efficiency. Kolikant (2009) found that many students accept as correct algorithms that no expert would accept. Thus, they do not force an algorithm to be valid on all input data, and they may even accept as correct an algorithm which is successfully compiled. Simon *et al* (2006) also found difficulties in students carrying out worst-case analysis of efficiency.

It is interesting to see how this weak misconception of optimality is shared with a strict misconception of optimality. Many students were unable to accept that several selection functions could be optimal. We wonder if they could be influenced by everyday situations where there is a strict ordering of elements, e.g. the table of a sport league or sorting an array.

The misunderstandings found do not seem to be specific of greedy algorithms, but of optimization problems in general. However, there is not a widely accepted subject matter in computer science curricula where basic concepts of optimization could be integrated. Related concepts of discrete mathematics, such as combinatory or probability have been identified in the ACM curricular recommendations (ACM, 2012), concept inventories (Almstrum *et al*, 2006), and prestigious algorithms textbooks that include preliminary concepts (Brassard and Bratley, 1996; Cormen *et al*, 2009). However, optimization concepts (e.g. maximization and minimization properties, or upper and lower bounds) are not addressed at all.

A minor observation is that we have found some behaviours that were reported in previous works (Simon *et al*, 2006), such as using real-world knowledge by students to elaborate their reasoning, changing the problem statement or the influence of examples provided on students' performance.

6.2 Removal of Misconceptions

We succeeded in removing the most severe misunderstandings and in reducing the misunderstanding of experimentation to a small percentage of students. We consider that such an enhancement was due to the interventions made, which were of four classes:

- **Classroom.** We gave emphasis on proposing selection functions and on the experimental analysis of their optimality. Underlying concepts regarding the scientific method were also explained and the teacher used the GreedEx interactive system to illustrate examples.
- **Assignments.** We introduced a preliminary session with a simple example to get familiarized with the experimental method and the GreedEx system. A previous improvement had been the incorporation of the last lab session to analyze and fix past wrong proposals.
- **Teaching materials.** Lecture notes were elaborated that explicitly address the new concepts and the most severe misunderstandings.
- **GreedEx system.** It was enhanced to ease the experimental method and the documentation of results. After the evaluations here described, we incorporated further enhancements (Velázquez -Iturbide, 2013; Velázquez-Iturbide *et al*, 2013), mainly to allow exporting tables for report documentation.

The enhancements introduced are heterogeneous, but constitute different ways of scaffolding the construction of students' knowledge of greedy algorithms. We intervened in all the instruction aspects we could identify. Classroom and assignment interventions were more or less obvious. The elaborated lecture notes were short in order to encourage their reading. The enhancements of the GreedEx system were based on usability evaluations (Velázquez-Iturbide *et al*, 2013) and were intended to remove any extraneous difficulty to the learning experience. Finally, it has often been advocated in the field of algorithm animation the importance of integrating teaching and systems (Bazik *et al*, 1998; Crescenzi and Nocentini, 2007), as well as including sessions to get familiarized with the target software.

We also made a secondary use of the cognitive conflict approach. Remember that students were urged, in the last session, to explain their failure to identify the optimal selection functions. Cognitive conflict has sometimes been criticized for denying students the validity of their results (Smith *et al*, 1994). However, we presented such a session to students as a second opportunity to do the assignment well.

6.3 Future Work

We hope to have contributed to the understanding of students' difficulties coping with optimization problems and their algorithmic solutions, especially in the scope of greedy algorithms. However, we feel that more about students' difficulties remains to be discovered. In particular, we do not know if students often have other misunderstandings about optimization problems. Furthermore, misunderstandings on other algorithm design techniques should be identified. We have noticed some misunderstandings based on anecdotal evidence, but more systematic studies are necessary.

Acknowledgments

This work was supported by research grant TIN2011-29542-C02-01 of the Spanish Ministry of Economy and Competitiveness.

References

ACM, INTERIM REVIEW TASK FORCE (2012): Computer Science curriculum 2008. ACM & IEEE Computer Society. Available at http://www.acm.org//education/curricula/ComputerScience2008.pdf.

ALMSTRUM, V.L., HENDERSON, P.B., HARVEY, V.J., HEEREN, C., MARION, W.A., RIEDESEL, C., SOH, L-K. and TEW, A.E. (2006): Concept inventories in computer science for the topic discrete mathematics. *SIGCSE Bulletin* 38(4):132–145. DOI= 10.1145/1189136.1189182.

BAZIK, J., TAMASSIA, R., REISS, S.P. and VAN DAM, A. (1998): Software visualization in teaching at Brown University. In *Software Visualization*. STASKO, J., DOMINGUE, J., BROWN M.H. and BLAINE, B.A. (eds), pp. 383-398. Cambridge, MA, USA, The MIT Press.

BEN-ARI, M. (2001): Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching* 20(1): 45-73. Available at http://www.editlib.org/p/8505/.

BRASSARD, G. and BRATLEY, P. (1996): *Fundamentals of Algorithmics*. Hertfordshire, UK, Prentice-Hall.

CLANCY, M. (2004): Misconceptions and attitudes that interfere with learning to program. In *Computer Science Education Research*. FINCHER, S. and PETRE, M. (eds), pp. 85-100. London, UK, Routledge.

CLEAR, T. (2004): Critical enquiry in CS education. In *Computer Science Education Research*. FINCHER, S. and PETRE, M. (eds), pp. 101–125. London, UK, Routledge.

CORMEN, T.H., LEISERSON, C.E., RIVEST, R.L. and STEIN, C. (2009): *Introduction to Algorithms*, 3rd ed., Cambridge, MA, USA, MIT Press.

CRESCENZI, P. and NOCENTINI, C. (2007): Fully integrating algorithm visualization into a CS2 course: A two-year experience. In *Proc. 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2007)*, pp. 296–300, ACM Press. DOI= 10.1145/1269900.1268869.

FINCHER, S. and PETRE, M. (eds) (2004): *Computer Science Education Research*. London, UK, Routledge. pp.3–4.

GAL-EZER, J. and ZUR, A. (2004): The efficiency of algorithms – misconceptions. *Computers & Education* 42(3): 215–226. DOI= 10.1016/j.compedu.2003.07.004.

GINAT, D. (2003): The greedy trap and learning from mistakes. *Proc. 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'03)*, pp. 11–15, ACM Press. DOI= 10.1145/792548.611920.

GLASER, B. and STRAUSS, A. (1967): *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Chicago, IL, USA, Aldine.

GÖTSCHI, T., SANDERS, I. and GALPIN, V. (2003): Mental models of recursion. *Proc. 8th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE'03)*, pp. 346–350, ACM Press.DOI= 10.1145/792548.612004.

HABERMAN, B., AVERBUCH, H. and GINAT, D. (2005): Is it really an algorithm – The need for explicit discourse. *Proc. 10th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE'05)*, pp. 74–78, ACM Press. DOI= 10.1145/1151954.1067469.

KLEINBERG, J. and TARDOS, É. (2006): *Algorithm Design*. Boston, MA, USA, Pearson.

KOLIKANT, Y.B-D. (2005): Students' alternative standards for correctness. *Proc. First International Workshop on Computing Education Research (ICER'05)*, pp. 37–43, ACM Press. DOI= 10.1145/1089786.1089790.

LINN, M. (1995): Designing computer learning environments for engineering and computer science: The Scaffolding Knowledge Integration framework. *Journal of Science Education and Technology* 4(2): 103–126. DOI= 10.1007/BF02214052.

MA, L., FERGUSON, J., ROPER, M. and WOOD, M. (2011): Investigating and improving the models of programming concepts held by novice programmers. *Computer Science Education* 21(1):57-80. DOI= 10.1080/08993408.2011.554722.

McCARTNEY, R., BOUVIER, D.J., CHEN, T-Y., LEWANDOWSKI, G., SANDERS, K., SIMON, B. and VANDEGRIFT, T. (2009): Commonsense computing (episode 5): Algorithm efficiency and balloon testing. *Proc. Fifth International Workshop on Computing Education Research (ICER'09)*, pp. 51–62, ACM Press. DOI= 10.1145/1584322.1584330.

McCAULEY, R., FITZGERALD, S., LEWANDOWSKI, G., MURPHY, L., SIMON, B. THOMAS, L. and ZANDER, C. (2008): Debugging: A review of the literature from an educational perspective. *Computer Science Education* 18(2): 67–92. DOI= 10.1080/08993400802114581.

MORENO, A., JOY, M., MYLLER, N. and SUTINEN, E. (2010): Layered architecture for automatic generation of conflictive animations in programming education. *IEEE Transactions on Learning Technologies* 3(2): 139–151. DOI= 10.1109/TLT.2009.36.

NORMAN, D. (1983): Some observations on mental models. In *Mental Models*. GENTNER, D. and STEVENS, A. (eds), pp. 7-14. Hillsdale, NJ, USA, Erlbaum.

ROBIN, A., ROUNDTREE, J. and ROUNDTREE, N. (2003): Learning and teaching programming: A review and discussion. *Computer Science Education* 13(2): 137–172. DOI= 10.1076/csed.13.2.137.14200.

SANDERS, I., GALPIN, V. and GÖTSCHI, T. (2006): Mental models of recursion revisited. *Proc. 11th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE'06)*, pp. 138–142, ACM Press. DOI= 10.1145/1140123.1140162.

SEPÄLÄ, O., MALMI, L. and KORHONEN, A. (2006): Observations on students misconceptions – A case study of the build-heap algorithm. *Computer Science Education* 16(3): 241–255. DOI= 10.1080/08993400600913523.

SIMON, B., CHEN, T-Y., LEWANDOWSKI, G., McCARTNEY, R. and SANDERS, K. (2006): Commonsense computing: What students know before we teach (Episode 1: sorting). *Proc. 2006 International Workshop on Computing Education Research (ICER '06)*, pp. 29–40, ACM Press. DOI= 10.1145/1151588.1151594.

SMITH, J., DISESSA, A. and ROCHELLE, J. (1994): Misconceptions reconceived: A constructivist analysis of knowledge in transition. *Journal of the Learning Sciences* 3(2): 115–163. DOI= 10.1207/s15327809jls0302_1.

TAHERKHANI, A., KORHONEN, A. and MALMI, L. (2012): Categorizing variations of student-implemented sorting algorithms. *Computer Science Education* 22(2): 109–138. DOI= 10.1080/08993408.2012.692917.

VELÁZQUEZ-ITURBIDE, J.Á. (2011): Resultados de tres sesiones de indagación experimental sobre algoritmos voraces. *Serie de Informes Técnicos DLSI1-URJC*, technical report 2011–04, Departamento de Lenguajes y Sistemas Informáticos I, Universidad Rey Juan Carlos, Spain. Available at http://www.dlsi1.etsii.urjc.es/doc/DLSI1-URJC_2011-04.pdf.

VELÁZQUEZ-ITURBIDE, J.Á. (2012): Refinement of an experimental approach to computer-based, active learning of greedy algorithms. *Proc. 17th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2012)*, pp. 46–51, ACM Press. DOI= 10.1145/2325296.2325311.

VELÁZQUEZ-ITURBIDE, J.Á. (2013): An experimental method for the active learning of greedy algorithms. *ACM Transactions on Computing Education*, 13(4): article 18, 23 pp. DOI=10.1145/2534972.

VELÁZQUEZ-ITURBIDE, J.Á. and DEBDI, O. (2011): Experimentation with optimization problems in algorithm courses. *Proc. International Conference on Computer as a Tool (EUROCON'11)*, 4pp. DOI= 10.1109/EUROCON.2011.5929294.

VELÁZQUEZ-ITURBIDE, J.Á., DEBDI, O., ESTEBAN-SÁNCHEZ, E. and PIZARRO, C. (2013): GreedEx: A visualization tool for experimentation and discovery learning of greedy algorithms. *IEEE Transactions on Learning Technologies* 6(2): 130–143. DOI=10.1109/TLT.2013.8.

VELÁZQUEZ-ITURBIDE, J.Á. and PÉREZ-CARRASCO, A. (2009): Active learning of greedy algorithms by means of interactive experimentation. *Proc. 14th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE'09)*, pp. 119–123, ACM Press. DOI= 10.1145/1595496.1562917.

WINSLOW, L.E. (1996): Programming pedagogy – A psychological overview. *SIGCSE Bulletin* 28(3): 17–22. DOI=10.1145/234867.234872.

Biographical Notes

J. Ángel Velázquez-Iturbide *received the Computer Science degree and the PhD degree in Computer Science from the Universidad Politécnica de Madrid, Spain, in 1985 and 1990, respectively. He is currently with the Universidad Rey Juan Carlos as a professor, where he is the Director of the Department of Computing Languages and Systems I and the head of the Laboratory of Information Technologies in Education (LITE). His research areas are educational software and educational innovation for programming education, software visualization, and human-computer interaction. Professor Velázquez is an affiliate member of IEEE Computer Society and IEEE Education Society, and a member of ACM and ACM SIGCSE. He is the President of the Spanish Association for the Advancement of Computers in Education (ADIE).*

J. Ángel Velázquez-Iturbide