GreedEx: A Visualization Tool for Experimentation and Discovery Learning of Greedy Algorithms

J. Ángel Velázquez-Iturbide, Member, IEEE Computer Society, Ouafae Debdi, Natalia Esteban-Sánchez, and Celeste Pizarro

Abstract—Several years ago we presented an experimental, discovery-learning approach to the active learning of greedy algorithms. This paper presents GreedEx, a visualization tool developed to support this didactic method. The paper states the design goals of GreedEx, makes explicit the major design decisions adopted, and describes its main characteristics in detail. It also describes the experience of use, the usability evaluations conducted, and the evolution of GreedEx in these years in response to the findings of the usability evaluations. Finally, the positive results obtained in an evaluation of educational effectiveness are shown. The paper has three main contributions. First, the GreedEx system itself is an innovative system for experimentation and discovery learning of greedy algorithms. Second, GreedEx is different from other visualization systems in its support to higher levels of learning, in particular evaluation tasks. Finally, GreedEx is an example of a medium-term research project, where an educational system was designed from explicit learning goals and was later refined in a user-centered design process involving instructors and students, before carrying out a successful evaluation of educational effectiveness.

Index Terms—Computer science education, learning via discovery, user-centered design, visualization systems and software

1 INTRODUCTION

Instructors and researchers have been interested in program and algorithm visualization for the last three decades [1]. As a consequence, a large number of visualization systems (and built-in animations) are currently available for educational use [2], [3].

If visualization systems were designed according to explicit educational goals, it would be easy to state the kind of tasks they support. However, hardly any visualization system has been designed this way. Alternatively, we may deduce their implicit goals by examining the literature. An analysis reveals that algorithm visualization systems are usually designed to aid students in understanding algorithm behavior [2], while program visualization systems are usually designed to aid in analyzing program behavior.

Our hypothesis is that the use of explicit pedagogical goals increases the quality and effectiveness of a visualization system. From the point of view of the developer, educational activities can be checked for alignment with the educational goals supported by the system.

In this paper, we exemplify this approach with a visualization tool intended to support a didactic method for greedy algorithms. Several years ago, an analysis of visualizations of algorithm design techniques [4] revealed a lack of common visualizations for the greedy technique. Further enquiry revealed that the treatment given to greedy algorithms in textbooks was deficient. Consequently, we proposed an innovative didactic method with explicit learning goals, based on experimentation and aimed at discovery learning [5]. Furthermore, we designed and developed a visualization system, called GreedEx, intended to support the didactic method. Two prototype antecessors of GreedEx were documented elsewhere [6], but in this paper we describe a mature and comprehensive version of GreedEx, as well as its underlying design principles, the evaluations conducted to enhance it, and its evolution. We also describe an evaluation of student performance conducted to check its educational effectiveness.

The paper has three main contributions. First, we present GreedEx itself, as an innovative system for experimentation and discovery learning of greedy algorithms. Second, GreedEx is different from other visualization systems in its support to higher levels of learning, in particular, evaluation tasks. Finally, GreedEx constitutes an example of a medium-term research project, where an educational system was designed from explicit learning goals and was later refined in a user-centered design process involving instructors and students, before carrying out a successful evaluation of educational effectiveness.
The structure of the paper follows. In Section 2, the experimental method is introduced. Section 3 gives an overview of GreedEx and states its design goals. Section 4 describes the GreedEx system in detail. Section 5 describes our experience using GreedEx and the usability evaluations conducted, and Section 6 overviews the enhancements introduced into GreedEx. In Section 7, we describe an evaluation of educational effectiveness. In the last two sections, we discuss related work, highlight our findings, and outline future work.

2 A Didactic Method for Greedy Algorithms Based on Experimentation

A typical set of educational goals for greedy algorithms, as well as the difficulties to achieve them, has been described elsewhere [5]. Our didactic method is intended to support a major learning goal: “State a selection function for a given problem and prove its optimality.” In summary, the didactic method asks the student to find out all the selection functions that could characterize an optimal greedy algorithm for a given optimization problem.

The didactic method has been presented elsewhere [5], but we reproduce it here in so that the reader may better understand the learning goals of GreedEx. For concepts on greedy algorithms, we refer to the excellent textbooks available (e.g., [7], [8]). Detailed recommendations about how to use the didactic method have also been elaborated [9], [10].

A selection function is defined on the set of available candidates, returning at each step the most promising candidate with respect to some measure. If we want to make explicit the design process of a selection function, we may first state a list of possible selection functions and then experiment with them for optimality. In general, the set of possible selection functions is not very large, as they are restricted to the values of some parameters or to values derived from them, sorted in increasing or decreasing order.

For instance, consider the activity selection problem [7]. Given a set of \( n \) activities, each one characterized by a start time \( s_i \) and a finish time \( f_i \), we seek a maximum-size subset of nonoverlapping activities. For instance, given the set of activities contained in Table 1, the subset \( \{7, 6\} \) is a valid solution, and subset \( \{7, 3, 10, 5\} \) is a maximum-size solution.

For this problem, we may state the following selection functions:

- Increasing order of start time (denoted \( S^\uparrow \)).
- Decreasing order of start time (\( S^\downarrow \)).
- Increasing order of finish time (\( F^\uparrow \)).
- Decreasing order of finish time (\( F^\downarrow \)).
- Increasing order of duration (\( D^\uparrow \)).
- Decreasing order of duration (\( D^\downarrow \)).

Given a set of possible selection functions, we may experiment with them and discard the nonoptimal ones. In other words, we just need to find one counterexample of the optimality of a selection function to discard it. After discarding any selection function, those remaining are plausible candidates for optimality.

For the example given above, solving it using the former six selection functions yields the results of Table 2.

As a consequence of this execution, the two selection functions that resulted in fewer activities (i.e., \( F^\downarrow \) and \( D^\downarrow \)) can be discarded. Assume that we randomly generate new input data. If we repeat the experiment, the four remaining selection functions are reduced to three (see Table 3, 2nd run). Further experimentation with additional input data could also produce optimal values for the three remaining selection functions (see Table 3, 3rd and 4th runs). Consequently, we might be tempted to conclude that there is empirical evidence about the optimality of \( S^\downarrow \), \( F^\uparrow \), and \( D^\uparrow \) as selection functions. Then, we should attempt building a formal proof of their optimality. We refer to [10] for a discussion about the integration into a course of the discovery of counterexamples and proofs of optimality.

Notice that experimentation alone allows discarding suboptimal selection functions, but it cannot be taken as a proof of optimality of the remaining selection functions. In particular, the selection function \( D^\downarrow \) is not optimal for this problem. Systematic experimentation reveals that it is optimal in about 98 percent of the cases.

Notice also that the experimental method needs not be restricted to greedy algorithms, but it can be used as a motivation to other design techniques for optimization problems (e.g., branch and bound or dynamic programming). In this case, the learner should discover, for these optimization problems, at least one counterexample of the optimality of every selection function. More details can be found in [9].

### Table 1

<table>
<thead>
<tr>
<th>( i )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_i )</td>
<td>14</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>24</td>
<td>24</td>
<td>12</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td>( f_i )</td>
<td>22</td>
<td>19</td>
<td>16</td>
<td>16</td>
<td>28</td>
<td>25</td>
<td>29</td>
<td>8</td>
<td>15</td>
<td>8</td>
<td>24</td>
<td>22</td>
</tr>
</tbody>
</table>

An Instance of the Activity Selection Problem

### Table 2

<table>
<thead>
<tr>
<th>Selection function</th>
<th>Selected activities</th>
<th>#activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S^\uparrow )</td>
<td>( {9, 3, 10, 5} )</td>
<td>4</td>
</tr>
<tr>
<td>( S^\downarrow )</td>
<td>( {4, 10, 3, 7} )</td>
<td>4</td>
</tr>
<tr>
<td>( F^\uparrow )</td>
<td>( {9, 3, 10, 5} )</td>
<td>4</td>
</tr>
<tr>
<td>( F^\downarrow )</td>
<td>( {6, 7} )</td>
<td>2</td>
</tr>
<tr>
<td>( D^\uparrow )</td>
<td>( {5, 7, 3, 10} )</td>
<td>4</td>
</tr>
<tr>
<td>( D^\downarrow )</td>
<td>( {11, 4} )</td>
<td>2</td>
</tr>
</tbody>
</table>

Results of Applying Different Selection Functions to the Activity Selection Problem

### Table 3

<table>
<thead>
<tr>
<th>Selection function</th>
<th>1st run</th>
<th>2nd run</th>
<th>3rd run</th>
<th>4th run</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S^\uparrow )</td>
<td>4</td>
<td>3</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>( S^\downarrow )</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>( F^\uparrow )</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>( F^\downarrow )</td>
<td>2</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>( D^\uparrow )</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>( D^\downarrow )</td>
<td>3</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Results of Applying Different Selection Functions over Several Input Data to the Activity Selection Problem
3 AN OVERVIEW OF THE GREEDEX SYSTEM

The GreedEx system was designed to support the experimental method described above. Our initial intention was to develop an educational system to support the experimental method for any greedy algorithm. However, we had to abandon this goal for the less ambitious goal of designing a system that only supports a fixed but extensible set of problems. This restriction is due to the huge difference in the visualization of the different greedy algorithms: We cannot find commonalities in the figures of greedy algorithms contained in textbooks, but they are always domain-specific [4].

Currently, GreedEx supports six optimization problems, namely the activity selection problem [7] and five knapsack problems: The fractional and the 0/1 knapsack problems [7], [8], 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 optimally solved with greedy algorithms.

3.1 Getting Started with GreedEx

Fig. 1 shows the user interface of GreedEx in a session with the activity selection problem (see Section 2). Three areas can be clearly distinguished (apart from the window title, the main menu, and the icon bar). The top area is the visualization panel, which graphically displays data. The lower left area is the theory panel, with two tabs: The problem tab contains the problem statement written in natural language (visible in the figure), and the algorithm tab contains a greedy algorithm written in Java-like pseudocode. Finally, the lower right area is the table panel, with four tabs containing the input data table, the results table, the history table (visible in the figure), and the summary table. The first three tables correspond to Tables 1, 2, and 3 in Section 2 above.

When the user launches the application and selects a specific problem, only the theory panel hosts visible contents. Its two tabs can be read by the students to understand the problem. Afterward, the student may generate input data from three sources: From the keyboard, randomly or from a file. Some limits are imposed on data size to keep it understandable for learning and manageable for visualization. Then, input data are displayed in the visualization panel and the student may choose any selection function available in the system for the active problem. To make the experiment a bit more complex and interesting, two additional selection functions are available for any problem (see Fig. 1): Increasing/decreasing order of index (respectively, denoted Ind ↑ and Ind ↓).

Every time the student chooses a selection function, he or she may execute it flexibly with four execution/animation controls: Step forward, full forward, step backward, and rewind. As animation controls are clicked, the visualization is updated accordingly. After a selection function has been completely executed, its results are stored in a row of the results table and in a cell of the history table. The summary table shows the percentage of optimal results obtained by each selection function in past executions.

Additional facilities are provided for faster experimentation: Executing all the selection functions over the current input data, executing a subset of the selection functions, and executing all the selection functions over a very large number of randomly generated input data. Finally, GreedEx includes a number of educational facilities.

GreedEx is available at the following URL: http://www.lite.etsit.urjc.es/greedex/.

3.2 Design Goals of GreedEx

GreedEx was built according to several pedagogical goals. Remember the pedagogical goal given above on selection functions: “State a selection function for a given problem and prove its optimality.” This goal was refined according to the more specific activities of the experimental method. Consequently, GreedEx must aid students with no background on greedy algorithms in achieving the following levels of Bloom’s taxonomy [11] for every greedy algorithm:

- **Comprehension level.** The student must understand the problem and the greedy algorithm that solves it.
- **Analysis level.** The student must be able to analyze the effect of applying a greedy algorithm to the same input data with different selection functions.
- **Evaluation level.** The student must be able to evaluate the outcome of different selection functions and select the optimal ones.

The system also had to satisfy several additional requirements for its educational use:

- **Easy to find, install, and execute** [12]. Instructors have little time and, thus, will only adopt GreedEx if it presents these advantages. The same applies to students with respect to their use out of lab time.
- **Easy to learn and use.** The student should be able to interact with GreedEx from the beginning with just a minimum understanding of the experimental method. Furthermore, the system functionalities must be discoverable with use. The rationale of this goal is reducing the cognitive load of the students using GreedEx to a minimum, so they can concentrate on the experimentation task. Finally, experimentation should be flexible so that students may follow
different paths for the same goal and backtrack to check previous results.

- **Motivating tool.** Obviously, the system must be visually appealing. Furthermore, motivation can be achieved by having the user engaged with the task and intrigued with respect to the experimental results.
- **Support for educational tasks** [12]. A number of options or functions must be incorporated to allow easy accommodation of the system to the instructor preferences or to support mundane educational tasks (e.g., documentation of experiments results).

## 4 Detailed Description of the GreedEx System

The experimental method described in Section 2 generates a large amount of data. To be able to handle these data effectively, they have been structured into three levels of detail. In this section, we make use of this structuring decision to describe in detail the main elements of GreedEx functionality. The first three sections describe these levels of detail, and then the connections between levels and other educational features are presented.

### 4.1 Single Set of Input Data, Single Selection Function

The smallest piece of experiment that can be carried out consists in executing the greedy algorithm for a particular input data set and a particular selection function. The learning goals of this activity are:

- **Comprehension level.** Understand the algorithm behavior with the given input data and selection function.
- **Analysis level.** Analyze the effect of the selection function.

The system gathers and displays data to support these learning goals. The visualization panel and the animation bar are the main means to support comprehension, while the results table is the main means to support analysis. We describe here animations and visualizations, and the description of the results table is delayed to the following section.

Input data are displayed at the visualization panel in a problem-specific format. The user may control the pace of execution with the animation controls. As the execution proceeds, the execution state is updated in the visualization panel. Upon termination of the algorithm execution, its outcome is updated in the results, history, and summary tables.

The animation bar mimics a basic set of VCR controls. There are four execution controls available: Step forward, full forward, step backward, and rewind. Notice the symmetry of controls for forward and backward directions, necessary to allow flexible tracing. For this level of detail, execution and animation are synonymous terms.

Currently, GreedEx provides two visualization formats, one for knapsack problems and another one for the activity selection problem. We reviewed illustrations in textbooks to adopt the best graphical representations. Consequently, our visualizations are variants of the figures contained in Sedgewick’s book for the knapsack problem [13] and in Cormen et al. [7] for the activity selection problem. In the latter, we were able to simplify the figure representation thanks to the time dimension provided by animation.

Fig. 1 illustrates a visualization for the activity selection problem in an intermediate state of execution. The activities are displayed in a bidimensional matrix with a temporal axis. Each activity is displayed as a horizontal bar, ranging between its start and finish times.

In general, visualizations must always embed the following information:

- **Input and output data.** If the output is a subset of the input, color coding is used to differentiate them in the same display, for example, in the activity selection problem. Otherwise, the visualization panel is split into two parts, for input and output.
- **Status of candidates (explained below).** Pending, discarded, selected.
- **Relative order of pending candidates with respect to the current selection function.**

Colors are used to make clear the status of each activity during the execution of the greedy algorithm. During the execution of a greedy algorithm, a candidate can be in one of the following states:

- **Pending.** It has not been selected by the selection function yet.
- **Selected.** It was selected and it was a valid candidate; therefore, it was inserted into the solution set.
- **Discarded.** It was selected but it was an invalid candidate; therefore, it was discarded.

In Fig. 1, the status of activities are, respectively, coded in colors blue, green, and red. For example, the only selected activity at the state displayed in the figure is activity 2 (in green).

The use of color tones allows sharing the same visualization for the different selection functions while still reporting the relative order of the candidates under each one. The criterion we have adopted is that the darkest tones are assigned to the candidates that are selected first, according to the active function selection.

In Fig. 1, the activities were selected in increasing order of start time, $S \uparrow$. Notice that four activities were considered (activity 2 starting at instant 1, and activities 6, 9, and 5, starting at instants 1, 2, and 3, respectively) and there are seven pending activities. The tones of pending activities become lighter as their left bounds are situated more distantly.

Fig. 2 contains an alternative assignment of tones for increasing order of duration ($D \uparrow$) as the selection function. Shorter activities are colored in dark, activity 10 being the shortest. Longer activities are colored in light, activity 0...
being the longest. Nevertheless, note the limits of human perception for distinguishing a high number of tones. To assist in this drawback, column “Order” at the left side of the figure shows the order of selection.

Visualizations contain redundant information that enhances their readability. At the left-hand side of the visualizations input data are shown in a matrix, textual format. Notice in Fig. 1 that five columns are shown for the activity selection problem. For each activity, we can read its corresponding index, start time, finish time, duration, and relative order under the current selection function. For instance, we see that activity 2 (in green) has start time 1, finish time 9, duration 8, and is in the 0th position (i.e., the first one) for the current selection function (increasing order of start time, $S \uparrow$).

Other redundant elements of the display are dependent on each problem. For example, Fig. 1 shows the use of dashed lines to project the time limits of each activity over the horizontal axis and a thick line placed on the horizontal axis to accumulate the time of the activities selected.

### 4.2 Single Set of Input Data, Multiple Selection Function

A higher level in the experiment comprises comparing the results of the different selection functions over the same input data. The learning goals of this level are:

- **Analysis level.** Analyze the effect of the different selection functions over the same input data.
- **Evaluation level.** Evaluate of the outcomes of the different selection functions and decide which are optimal for these input data.

The results table is the main means to display information relevant for analysis and evaluation. It resembles Table 2 (see Section 2), showing the results yielded by the different selection functions, executed over the same input data set. The table has one row per selection function. When the algorithm is executed for a particular selection function, the corresponding row is filled in. Some columns are common for any problem:

- Selection function.
- Candidates selected under such a function.
- Outcome produced by these candidates.

In some problems, additional columns are useful to better understand the selection of candidates and their effect. For instance, in the knapsack problem, two additional columns store the value associated to each candidate according to the selection function, and the ordering of candidates under this selection function.

A set of execution functions allows the user to think of execution at a higher level of abstraction than the animation bar. Two execution functions allow selecting and executing, in a single step, the next or the previous selection function over current input data, respectively. As a result, the visualization panel displays the final state of its execution, and the results table updates its result as well.

At an even higher level of abstraction, two additional execution functions allow executing in a single step all or a subset of all the selection functions over current input data. The result table is completely filled as a result of any of these atomic actions.

### 4.3 Multiple Set of Input Data, Multiple Selection Function

At the highest level of the experiment, the selection functions must be compared for optimality. The learning goal of this level is:

- **Evaluation level.** Evaluate the outcomes of selection functions and decide which are optimal.

The user should be able to consult the results of executing the different selection functions over the current and past input data sets. Consequently, he or she should be able to visually recognize the selection functions that always yield optimal results. These data are displayed in the history table, which is similar to Table 3 (see Section 2), with its axes switched. Each row of the history table stores the results obtained for a particular data set. Each column stores the results obtained for a selection function. Each cell may contain either a numeric value or a dash, depending on whether the corresponding selection function has been executed with the corresponding input data.

To facilitate the user to identify optimal selection functions, two highlighting effects are used. For each row, cells that contain optimal outcomes have their background highlighted in gray. In addition, the columns that contain optimal values in all of its rows show the font color of these values highlighted in light blue.

Fig. 1 shows the history table in the lower right panel. Notice that only two selection functions had always produced optimal results in the experimentation illustrated (namely $E \uparrow$ and $S \uparrow$).

Two additional elements are provided in GreedEx for an even higher level of abstraction. They are most useful for problems where a high number of executions are necessary to obtain high evidence on the results, e.g., the activity selection problem.

The “intensive execution” function allows the user, in a single step, to generate a high number of input data and execute all the selection functions over them. The user may either provide the number (in hundreds) of input data sets to generate or the time (in seconds) to spend in this generation-execution process.

Finally, the summary table shows the percentage of executions for which each selection function was optimal. It only considers those input data, where all the selection functions have been executed.

### 4.4 Connection between Levels

Experimentation rarely follows a strictly sequential process. In general, the user examines data in forward and backward directions. We briefly mentioned this fact above: The student often needs to backtrack an animation to understand a previous step.

The need to watch past results also arises at higher levels of detail. For instance, the student may want to watch the past behavior of any selection function over the current data set. Moreover, he or she may want to watch that behavior over a past data set. To support such exploration, the user must be able to backtrack using adequate affordances. Fortunately, a very intuitive affordance for these connections consists in simply clicking on a row of the results table or on a cell of the history table. Then, the visualization panel
is refreshed with the corresponding input data and selection function, displaying the final state of its execution. Now, the student may backtrack in the animation or replay it for his or her inquiry.

Fig. 1 illustrates the effect of this flexible connection. After generating 100 input data sets and executing over them, the display in the visualization panel corresponds to the fifth input data set and to selection in increasing order of start time, S ↓. The user clicked the corresponding cell in the summary table, then rewound the visualization, and advanced four steps forward, obtaining the display shown in the figure.

### 4.5 Educational Support

GreedEx incorporates a number of functions to better support teaching and learning. One critical issue is system adoption by instructors [12]. To this end, the user interface of GreedEx was developed to support internationalization. A translation XML file allows specifying equivalences between languages. Currently, it supports English and Spanish.

A second useful feature consists in allowing the instructor to customize visualizations and animations to his or her teaching style. GreedEx allows configuring the colors of the three kinds of input candidates, as well as enabling or disabling in animations a flickering effect over the next candidate.

Third, some instructors may consider, from a didactical point of view, that students should conduct the experimental process in a more directed way. Consequently, GreedEx supports a restricted use of execution functions:

- For the first input data generated, the system only allows executing each selection function individually, either step-by-step or atomically.
- For the second input data, GreedEx also allows executing all or a subset of the selection functions over these input data.
- For the third and subsequent executions, it also allows intensive execution.

Last but not least, GreedEx supports documentation tasks. Students will probably be required to experiment and write a report describing their findings. GreedEx provides a number of functions to export visualizations, animations, and tables.

### 5 EXPERIENCE AND USABILITY EVALUATION

GreedEx evolved in past years. We conducted several kinds of evaluations that, following a user-centered design, allowed fixing errors, enhancing existing functions, and supporting missing features. In this section, we first summarize the use of GreedEx in actual teaching, and then we present the evaluations conducted.

#### 5.1 Experience of Use

We have used GreedEx for five academic years (since the academic year 2007/2008) in the first quarter of the mandatory third-year course "Design and Analysis of Algorithms" offered at our University to Computer Science students. We have been using GreedEx both in the classroom and in the laboratories.

In the classroom, the two first sessions on greedy algorithms were slightly changed to give more relevance to the identification of optimal selection functions. The instructor used GreedEx as a quick means to illustrate variability in selection functions and their differences with respect to optimality.

The course schedule included one laboratory session for an assignment on greedy algorithms. This assignment was changed from a traditional coding assignment to an experimental, discovery assignment. Its format changed slightly over the years, according to our experience. More detail about the lab sessions is given below in Section 5.3.1.

We want to remark here that the assignment is a learning-by-discovery activity. The results are not known in advance by the user, and their discovery is an engaging activity. It is noteworthy that even the developers of GreedEx obtained unexpected results. As explained by Velázquez-Iturbide and Debdi [9], two equivalent selection functions can be stated for the knapsack problem. However, textbooks only report one of them (that with the most intuitive statement). Analogously, Cormen et al. [7] only identify one optimal selection function for the activity selection problem, but GreedEx allows identifying two symmetric selection functions ($S$ and $F'$), equally intuitive. Furthermore, GreedEx allows identifying a nearly optimal selection function ($D$), which could be used as a very good heuristic.

#### 5.2 Usability Evaluations by Experts

We conducted expert evaluations [14], also called inspection methods [15] or analytic evaluation techniques [16]. Expert evaluations were conducted in the initial stages of GreedEx and also every time that substantial changes were introduced. They were mainly conducted by the first author, as a humble expert on the teaching of algorithms. The nature of an expert evaluation varied for each session, but it always fit into at least one of the following categories:

- **Requirements testing** [17]. These evaluations tried to check that a new system release satisfied the specifications and that it did not contain bugs. Therefore, their goal was to ensure correctness.
- **Heuristic evaluation** [18]. These evaluations tried to check that the user interface conformed to good principles (e.g., homogeneity). Therefore, their goal was to ensure ease of use and satisfaction.
- **Cognitive walkthrough** [19]. These evaluations tried to check that the tool supported the experimental process well. Therefore, their goal was to ensure ease of use and task effectiveness.

Typically, the expert either had a list of changes to be checked or was to perform some tasks, representative of typical scenarios. The result of each expert evaluation was a list of items marked (either as done, partially done, or pending), as well as a list of new changes to be introduced.

In addition, the tool was used by the expert in a range of on-the-fly situations: In-class, laboratory, demos, and preparation of research papers. These situations often required the tool to be effective and efficient in performing specific tasks; thus, they can be considered informal kinds of cognitive walkthrough evaluations. Difficulties
encountered in these situations were written in notes or just remembered, and later replicated with GreedEx and documented by the expert.

We have not counted the number of expert evaluations, but it probably is over 100. As a consequence, the authors felt very confident about the adequacy of GreedEx to its intended educational goals. The main drawback of expert evaluations is that their results do not necessarily generalize to final users. In our case, the results may not generalize to other faculty or to students. Therefore, we initiated other actions. With respect to faculty, educators from the same department were encouraged to use the tool for their algorithms courses. Two instructors used it in different academic years and reported satisfaction with the tool. We only obtained one generic suggestion from one instructor, which was the seed for the design of the intensive execution function. With respect to students, we conducted the usability evaluations described in the following section.

5.3 Usability Evaluations Based on Questionnaires

As we explained in Section 5.1, the course schedule included one laboratory session for an assignment on greedy algorithms. We made use of this session to conduct five usability evaluations with students, based on questionnaires [16], also called surveys [12]. In the first section, we describe the evaluation protocol, and in the following two sections we present the results obtained.

5.3.1 Procedure of Evaluations Based on Questionnaires

We conducted five evaluations, held in January 2008, May 2008, January 2009, November 2009, and November 2010, respectively. The evaluation conducted in May 2008 corresponds to the optional course “Advanced Algorithms and Data Structures,” also offered to third-year students.

The procedure was similar for all the evaluations (although some changes were introduced for the fourth and fifth evaluations). Students downloaded all the materials at the beginning of the session: the assignment statement, GreedEx, and a report template. In addition, they were given an opinion questionnaire on a sheet of paper. The assignment statement contained the problem statement and a short description of GreedEx. They had to perform three tasks:

1. Use GreedEx to determine the optimal selection functions for a problem. In the first evaluation, the knapsack problem was used, but in successive evaluations, the selection activity problem was preferred because of the higher difficulty of showing the nonoptimality of the D ↑ selection function.
2. Complete and electronically deliver a short report, following a template with a simple outline.
3. Fill out and deliver the opinion questionnaire (optional).

The assignment could be solved either individually or in pairs, but the questionnaire had to be filled out individually. We measured ease of use, user’s satisfaction, and task effectiveness. The questionnaire consisted of three parts: multiple-choice questions on general issues, multiple-choice questions on specific elements, and open questions on general issues. Answers to multiple-choice questions were in a Likert scale ranged from 1 (very bad) to 5 (very good). The number of questionnaires gathered in these evaluations was 40, 11, 28, 27, and 36, respectively.

5.3.2 Results to Multiple-Choice Questions

We asked students to give their opinion in four multiple-choice questions, directly related to ease of use, effectiveness, and satisfaction. Table 4 shows the questions and the results obtained. The questions are placed in rows. The first column, labeled “Avg.,” contains the average value of the students’ answers collected in the five evaluations. The following five columns, labeled “M.n,” contain the median of students’ answers collected in the nth evaluation. The median is a most adequate measure for ordinal values, but we show the mean for a more intuitive perception of students’ opinions.

Notice that very high values were obtained, especially for ease of use (mean 4.44, row 1). The mode for this criterion was 5 in all the evaluations but the fifth one. Students also gave very high ratings to effectiveness (4.32 and 4.16, rows 2 and 3) and satisfaction (4.24, row 4). Satisfaction and effectiveness for analyzing selection functions had regular ratings, whereas effectiveness for identifying optimal selection functions suffered more ups and downs. This fact is due to the changing number of functions that GreedEx incorporated in the successive evaluations (see Section 6).

Although it is not shown in Table 4, it is interesting to note the only mean that was obtained under 4 regarded effectiveness for identifying optimal selection functions (3.60, second evaluation). This mark was due to the limited set of facilities that GreedEx had in that evaluation for nontrivial problems (such as the activity selection problem). This mark increased in the third and following evaluations thanks to the inclusion of the history table in GreedEx.

The second part of the questionnaire measured the perceived quality of the main elements of GreedEx. Table 5 is labeled similarly to Table 4, with the main elements of GreedEx in rows. The elements of GreedEx are presented sorted in descending order of their global mean. Some cells do not have a value because either that element did not exist at the time of the evaluation or the questionnaire did not include it.

The table shows that their values are very high for the most relevant elements of the user interface for interactive experimentation (mean 4.10 or higher). Most elements with lower values (3.98 or less) are not critical for experimentation (e.g., interactive help) or are the most static elements.
(e.g., theory tabs). These results are probably due to the fact that we have (unconsciously) dedicated more attention to the key features of GreedEx for experimentation (including the main menu).

5.3.3 Results to Open Questions

We urged students to give their opinion, in free format, about five issues: Positive features of GreedEx, negative features, features difficult to use, useless features, and features that GreedEx did not have but it should. We used the students’ answers as a rich source of information to enhance specific elements of GreedEx.

Table 6 gives an overview of the answers given by students to open questions. Remember that the total number of gathered questionnaires was 142. Each row corresponds to an open question. They are presented top-down in decreasing order of students’ contributions.

Notice that about two-thirds of students identified positive elements in GreedEx, about one-half identified elements to include in GreedEx, less than 40 percent identified negative elements, and less than one-quarter identified elements difficult to use or useless. Therefore, the overall students’ impression about GreedEx was very positive, although they often had suggestions for improvement, mainly about elements to include into GreedEx. Moreover, they had relatively few suggestions about elements difficult to use or useless.

For a more detailed analysis of the answers, we performed an iterative analysis and classification of the answers into categories, coming to a final classification into nine categories:

- **General comments.** These are comments on GreedEx as a whole, expressed as subjective impressions or technical judgments.
- **Visualization and animation.** They usually refer to the visual presentation of data, but they may also refer to the animation, the use of colors, or customization. Answers that refer to the step-by-step execution are classified here, as the user must perform this execution with the animation controls.
- **Selection and execution of different selection functions.** These answers refer to the selection of selection functions or to the different execution functions (except the step-by-step execution).
- **Comparison of results, and tables.** They refer in general terms to the comparison of results and more specifically to the tables. We also include here comments about determining the optimality of selection functions.
- **Production of input data.** These answers refer to the random generation, the reading from a file or the interactive input of data.
- **Export.** They refer to different ways to export into files.
- **Other elements.** They refer to other features of GreedEx: The interactive help, the theory panel, icons, internationalization, or exit.
- **No comments.** Instead of leaving a cell blank, some students explicitly say that they have nothing to answer.
- **New elements.** They identified elements that GreedEx did not have, but it should incorporate in the future.

We may consider that each student gave to a given question: no answer, a simple answer, or a compound answer. A compound answer is an answer that can be divided into parts that fit into several categories. For instance, a student (evaluation 1, student 28) identified the following positive elements: “The clarity and simplicity of the program, the graphics, and especially the results table are very useful to me.” This opinion contains a general comment about the system as well as claims about visualizations and tables.

Table 7 contains a detailed account of the answers given to the five open questions. Notice that “general comments” are only given to the questions on positive or negative elements. Likewise, the answer “no comments” does not occur in the question on positive elements. Obviously, answers classified as “new elements” only occur in the question on elements to include.

The open question on positive elements was the question with the highest number of simple answers. Of 142 students, 95 (66.90 percent) answered this question, with a total of 119 simple answers. In general, answers to this question are written in a very positive mood.

We review the three most frequent categories, including some representative answers of students:

- **General comments.** Students consider that GreedEx is pleasant, easy to use, or useful. Some representative answers are: “Good graphical interface, easy, simple, and intuitive” (evaluation 1, student 1), “Easy, so it assists us in learning” (evaluation 5, student 7).
- **Visualization and animation.** These are very popular features of the system, as the following answers show: “It has been easier for me determining the...
intervals displayed as graphics rather than as pure arrays. This helps in thinking faster on the problem” (evaluation 2, student 7), and “I consider that it is very useful to be able to execute step-by-step because it has helped us in understanding the different selection functions” (evaluation 4, student 10).

Comparison of results and tables. Some answers follow: “Ease to compare data obtained with different selection functions” (evaluation 1, student 7), and “It allows seeing what selection functions are optimal among all for solving the problem” (evaluation 4, student 17).

The answers to the other four open questions were the main source of information about how to enhance GreedEx. They varied greatly in the five evaluations, depending on the features of the version used of GreedEx (see Section 6).

The answers on elements to include are mainly about enhancements of the visualization format, better support for tables or for exportation, and new elements. We do not go into details here about the enhancements introduced into GreedEx, as an overview can be found in Section 6. However, it is interesting to briefly comment on some enhancements.

Initially, we provided functions to export results in text format. Gradually, we removed this way of exporting and incorporated new functions to export into graphics files: One or a collection of visualizations, an animation, or a table.

A number of suggestions of new elements could not be considered. Some suggestions required substantial development effort, for instance, a dynamic code tab, coordinated with the animation state. Furthermore, other suggestions did not fit into the philosophy of the tool, such as allowing the user to define his or her selection functions or greedy algorithms.

The answers to the question on negative elements affected all the categories, and they allowed introducing many minor enhancements. The answers to the two remaining questions also were regularly distributed. It is worth noting the high percentage of students who gave a “no comments” answer (about 50 percent of the simple answers).

6 Evolution of GreedEx

The evaluations of GreedEx allowed gathering much information about its drawbacks. Consequently, it evolved substantially in these years.

Table 8 shows in detail the evolution suffered by the different elements of GreedEx. Each row contains an element of the system. The table has five columns, labeled with the name of a system and possibly a version number. AMO and SEDA are the names of two antecessors of GreedEx [6], each one aimed at experimenting with a single problem: AMO supported the knapsack problem, and SEDA supported the activity selection problem. They were used as follows in the five usability evaluations:

1. AMO.
2. SEDA v.1.
3. SEDA v.2.
4. SEDA v.2.
5. GreedEx v.1

The current version of GreedEx is number 2.

The cell contents of Table 8 must be interpreted as follows: The initial version of a system element is always marked as X. Successive minor changes in the element are denoted X, X, and so on. A major change in an element is denoted by a change of letter: Y, Z, and so on. GreedEx generalizes AMO and SEDA; therefore, it sometimes supports features of both systems. This generalization is denoted by a juxtaposition of their respective features, e.g., XY. Finally, the results table in GreedEx presents new columns for some of the additional problems feature denoted by a new occurrence of Z.

Many minor details were changed and enhanced, as Table 8 illustrates. Furthermore, the following functions were incorporated to the original design:

- Tables. Initially, GreedEx only provided two tables [6], namely the input data table and the results table. It was later considered convenient to include the history table and the summary table.
- Execution. Initially, GreedEx only allowed choosing and executing selection functions individually in the forward direction [5]. Afterward, the backward direction was included in animations, and new execution functions were included for higher speed: Select and execute the next or the previous selection function, execute all or a subset of all the selection functions with the active input data, and execute all the selection functions over a very high number of input data randomly generated.
Export. A number of functions to export visualizations, animations, and tables into graphics files were developed.

Internationalization. The user interface of GreedEx was redesigned so that it can be customized to any natural language. Currently, it supports English and Spanish.

Two additional elements that received much attention in expert evaluations were the menu structure and the icon bar. They were key means for keeping the user interface simple while the number of functions supported by GreedEx increased. The scores on ease of use obtained from students (see Table 4) show that we succeeded in keeping interaction simple.

### 7 Evaluation of Educational Effectiveness

We wanted to evaluate not only whether GreedEx was usable but also whether it was useful. Therefore, we conducted a controlled experiment to measure the effect of GreedEx on students’ performance. In this section, we describe the population sample and the procedure, as well as the results obtained.

#### 7.1 Population and Procedure

The participants were computer science majors enrolled at our university in a second-year mandatory course on design and analysis of algorithms. We used two enrolment groups; therefore, the mapping between students and groups was independent from the experiment (but it was not random). The control group was formed by 18 students enrolled in the campus of Vicálvaro of our university, while the experimental group was formed by 50 students enrolled in the campus of Móstoles.

Our experimental method is mainly intended to be used in the introduction to greedy algorithms. Therefore, the treatment of both groups only differed in the first, second, and fourth sessions of this chapter. All the sessions were two hours long. In the first session of the experimental group, emphasis was given not only to the optimal selection function for some simple problems but also to the proposal of alternative selection functions. In the first half of the second session, the instructor performed a demonstration of GreedEx to students, and in the second half, students had to use it at the computer laboratory to find the optimal selection functions for the knapsack problem, which is a simple challenge. The purpose of this lab session was to make students become familiar with the experimental method and the GreedEx system. Finally, in the fourth session students had to solve an assignment in the computer laboratory consisting in finding the optimal selection functions for the activity selection problem, which is a much more demanding problem.

Students in the control group followed a traditional schedule. The contents of the three sessions were similar but without any reference to the experimental method or GreedEx. In particular, the assignment was a traditional assignment consisting in designing and coding an optimal greedy algorithm for the activity selection problem.

The independent variable in the evaluation was the use of the experimental method and GreedEx for learning greedy algorithms. The dependent variable, which we call TEST, is defined as the subtraction of grades obtained by students in the posttest and the pretest. Both tests were
However, the experimental group shows a remarkable improvement of grades, obtaining a mean of 6.32 and a standard deviation slightly greater than in the pretest (1.58).

Comparing pretest versus posttest, after analyzing data, we did not conclude normality for the control group (obtaining \( p < 0.05 \) of significance using the Shapiro-Wilk test for both groups), so we decided to perform the Wilcoxon test for nonparametric samples. In this test, we assume that the null hypothesis can be stated, as there are no differences between the medians, so a \( p \)-value greater than 0.05 will reveal homogeneity in the samples. As a result, we obtain that there are no significant differences between both grades (\( p = 0.556 \)). In other words, after explaining greedy algorithms with traditional means to the control group, it cannot be concluded that there is any improvement in the test grade.

The experimental group was formed by a higher number of participants. The presence of normality for the pretest and the posttest (obtaining in both cases significance of \( p = 0.200 \) with the Kolmogorov-Smirnov test), as well as significant correlation between samples (\( r = 0.386, p = 0.006 \)) allowed using the Student t-test for related samples. The differences found between the pretest and the posttest grades were very significant (\( t = -5.028, p = 0.000 \)). Consequently, it was deduced that the participants had a significant improvement in the test grades after using the GreedEx system. However, this improvement was not experienced by the control group.

To obtain an additional information about the magnitude of the change produced by the different learning approaches, we computed the size of the effect and the percentage of change in both groups by means of the \( d \) statistics of Cohen [20]. Table 10 shows that no change can be appreciated in the control group. However, the experimental group delivers a value \( d = 0.79 \), corresponding to a medium effect, very close to a large effect (it is considered large for \( d = 0.80 \) or higher). Likewise, a percentage of change close to 22 percent shows a medium level of change for the experimental group.

In summary, the results of our experiment show that the use of GreedEx had a positive influence on students' performance for learning greedy algorithms, obtaining significant differences in the grade used to assess their knowledge level before and after using GreedEx. However, there is no significant improvement in the control group, where greedy algorithms were taught with traditional means. Furthermore, the results of effect size are very relevant, counting a medium-high value as a consequence of using GreedEx.

### 8 Related Work

Our work has several facets, from issues specific of greedy algorithms to general issues on experimental inquiry. In this
section, we discuss other educational systems relevant to these facets and their relation to GreedEx.

### 8.1 Systems Supporting Greedy Algorithms

We hardly find systems supporting the teaching or learning of greedy algorithms. In a survey on visualizations of algorithms, classified by algorithm design techniques, it was found that there are no common visualizations for the greedy technique [4]. Consequently, it is not surprising that no general systems can be found to support this technique.

We find, however, some systems for learning specific greedy algorithms. For instance, Sánchez-Torrubia et al. [21] present the interactive tutor PathFinder aimed at supporting Dijkstra’s algorithm. We also find a myriad of custom animations constructed for specific greedy algorithms. For instance, the comprehensive portal of algorithm animations Algoviz [3] contains 15 animations of Dijkstra’s algorithm, 15 of Kruskal’s or Prim’s algorithms, 13 of Huffman codes, and 3 animations of other greedy algorithms. Some of them were developed from scratch by instructors, while others were developed using well-known algorithm animation systems, such as ALVIE [22], ANIMAL [23], JHAVE [24], or Trakla [25].

Our system is in an intermediate position in a range varying from general systems to problem-specific systems. GreedEx supports a fixed set of problems (currently, six problems), but it can be extended to other problems at little effort. Moreover, for each basic greedy algorithm, GreedEx visualizes a set of different selection functions.

With respect to the visualizations themselves, all the systems (including GreedEx) deliver custom visualizations. However, GreedEx uses the same basic visualization for all the selection functions defined for a given algorithm. The visualization is adapted to each particular selection function by using color tones to implicitly sort the problem candidates. In general, this technique can be used for any greedy algorithm.

An additional important difference is the level in Bloom’s taxonomy that the system supports. PathFinder or the custom animations cited above are aimed at understanding or analyzing optimal greedy algorithms. However, GreedEx is intended to also support the evaluation level of learning. Both optimal and suboptimal selection functions are supported, so students may make a reasoned choice of the optimal selection functions.

### 8.2 Systems Supporting Experimentation with and Visualization of Algorithms

Our experimental method is related to other experimental approaches to learning. Experimentation has a long tradition in the sciences education, and we lately find increasing interest on experimentation in computer science education [26]. If we focus on algorithms, we must first make clear the property to investigate. Experiences with algorithm experimentation deal with either correctness or efficiency, but we have not found any system supporting experimentation with optimality [27].

Advanced support for experimentation with efficiency includes comparison of the performance of alternative algorithms for the same problem or comparison of the performance of different cases for the same algorithm. These features can be found in some benchmarking [28] or algorithm animation [29] systems. The comparison is typically based on brute performance data [28] or is left to the user’s visual perception (e.g., end time of animations [29]). GreedEx allows comparing optimality results, relying on both optimization values and the visual perception of the user (highlighting effects in the history table).

Another useful feature of GreedEx is that it structures data and visualizations at different levels of abstraction. The few animation systems that give structure to visualizations work at the level of a single algorithm, for instance, the HalVis [30] or the “Algorithms in Action” [31] systems.

Finally, we can hardly find visualization systems developed using user-centered design. In general, they are not evaluated with respect to usability or they are evaluated with a single method [32]. In particular, details about either the evaluation procedure or the results are often missing in the literature.

### 8.3 Systems Supporting Discovery Learning

Experimentation provides an approach to discovery learning. This active-learning approach may be used at different educational levels [33] and may adopt different forms, but the student is always challenged to discover something that is not obvious in advance. Herron [34] proposes four levels of scientific inquiry (confirmation/verification, structured inquiry, guided inquiry, and open inquiry). Our proposal is a case of structured inquiry.

We remark that some of the results that can be obtained with GreedEx were not known in advance by the authors, as we explained in Section 5.1 (see also [9]).

Tools supporting discovery learning have been called “cognitive tools” by some authors [35]. A more general formulation leads to the concept of “creativity-support tools” [36]. According to Shneiderman, these tools must support exploratory search, allow for collaboration, keep a wide record, and allow design “low thresholds, high ceilings and wide walls.” As usability evaluations show, GreedEx has “low thresholds,” but its domain is very restricted, so it unfortunately has “low ceilings.”

### 9 Conclusion

We have described the visualization system GreedEx, constructed to promote active learning of greedy algorithms, as well as several features of GreedEx that make it unique in computer science education. First, it is intended to support an innovative experimental, discovery-learning method for the active learning of greedy algorithms. A minor contribution is a general technique, based on color tones, to share any visualization for a specific greedy algorithm among the different selection functions. Second, GreedEx was developed from explicit principles, both pedagogical and technical. In particular, it intends to achieve high-level pedagogical goals, namely evaluation goals. Finally, GreedEx was refined in a user-centered design process to guarantee its usability and its adequacy to the intended educational goals. We conducted a high number of usability evaluations involving different evaluation methods, in a number of different scenarios and with different final users. We also evaluated its educational...
effectiveness, obtaining statistically significant improvements in student performance.

In perspective, we made several choices to develop GreedEx that proved to be successful. Therefore, we may highlight three lessons learned. First, we wanted to develop an original system for a class of algorithms (i.e., greedy algorithms) lacking general visualizations. We started developing systems for specific problems and gained experience with them. As a consequence, we were able to extract their main features and to develop a more general system supporting several algorithms.

Second, successive usability evaluations allowed enhancing the system and incorporating missing functions. The participation of both experts and end users was a key element for the detection of improvement opportunities. We also dedicated much attention to keep the user interface simple, which proved to be a key element in students’ acceptance.

Finally, we only conducted an evaluation of educational effectiveness after conducting several usability evaluations. Consequently, the results of the effectiveness evaluation were not distorted by usability lacks of the system.

In the near future, we plan to further extend these efforts. First, GreedEx is being extended to support collaborative discussion. Collaboration makes special sense in the final phase, when students have to elaborate and deliver their proposal of optimal selection functions. Second, GreedEx support for discovery learning is limited to a small number of problems. An open challenge consists in designing and constructing a similar but more general tool, usable in a broader domain of optimization problems.

ACKNOWLEDGMENTS
This work was partially supported by the Spanish Ministry of Economy and Competitiveness under grant TIN2011-29542-C02-01. The authors want to thank all the colleagues who helped them in these years and also the constructive comments of the referees.

REFERENCES


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 a professor at the Universidad Rey Juan Carlos, where he is the director of the Department of Computing Languages and Systems I and the leader of the Laboratory of Information Technologies in Education. His research areas are software for and innovation in programming education, software visualization, and human-computer interaction. He is an affiliate member of the IEEE Computer Society and the IEEE Education Society and a member of the ACM and the ACM SIGCSE.

Ouafae Debdi received the management computing degree from the Universidad Rey Juan Carlos, Madrid, Spain, in 2007, and two MSc degrees in computer science and statistics in 2008 and 2009, respectively. She is working toward the PhD degree at the Universidad Rey Juan Carlos and is currently a fellow researcher at the Laboratory of Information Technologies in Education. Her research interests include human-computer interaction and software for programming.

Natalia Esteban-Sánchez received the computer science degree and the MSc degree in engineering systems decision from the Universidad Rey Juan Carlos, Madrid, Spain, in 2009 and 2010, respectively. She is currently working toward the MSc degree in interactive computing and multimedia. Her research interests include software for programming education and human-computer interaction.

Celeste Pizarro received two MSc degrees in mathematics and statistics in 2000 and 2001, respectively, from the Universidad de Extremadura, Badajoz, Spain, and the PhD degree in computer science and mathematical modeling from the Universidad Rey Juan Carlos, Madrid, Spain, in 2006. She is currently an assistant professor in the Department of Statistics and Operations Research at Universidad Rey Juan Carlos. Her research interests include different mathematical programming fields (stochastic, integer, linear) and their applications.