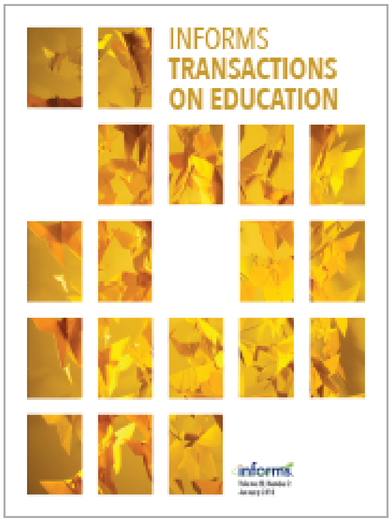


INFORMS Transactions on Education

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Teaching Heuristic Methods to Industrial Engineers: A Problem-Based Experience

Jesús Muñuzuri, Pablo Cortés, Rafael Grosso, Luis Onieva

Vol. 16, No. 2, January 2016, pp. 60–67
ISSN 1532-0545 (online)

INFORMS Transactions on Education

informs

http://dx.doi.org/10.1287/ited.2015.0152

Teaching Heuristic Methods to Industrial Engineers: A Problem-Based Experience

Jesús Muñuzuri, Pablo Cortés, Rafael Grosso, Luis Onieva
School of Engineering, University of Seville, 41092 Sevilla, Spain {munuzuri@us.es, pca@us.es, rgrosso@us.es, onieva@us.es}

Because of its mathematical and computational components, operations research (OR) is not simple to teach or to learn, despite its innumerable industry applications. However, advanced OR is included in many graduate degrees related to industrial engineering, where students need these techniques to solve complex optimization problems. Faced with the problem of teaching heuristic methods to master's students at the University of Seville, we developed a problem-based approach whereby instead of listening to lectures and taking exams on these techniques, one algorithmic technique is randomly assigned to each student, who must apply it to solving a certain optimization problem. Here we discuss our approach to putting our heuristics course into practice, the problems we faced, how we addressed those problems, the positive results obtained, and the lessons learned.

Keywords: teaching optimization; teaching with projects; active learning; heuristic methods; graduate course
History: Received: May 2015; accepted: November 2015.

1. Introduction: Teaching Operations Research

Basic optimization is present in many engineering curricula, usually covering linear programming (LP) problems, the Simplex method to solve those problems, use of spreadsheet applications to run Simplex automatically, and basic modeling concepts. These are the typical contents covered in an engineering undergraduate OR course (Strayer 1989, Hillier and Lieberman 2001, Winston et al. 2003). On the other hand, graduate OR contents focus on larger, more complex optimization problems that cannot be approached with a basic tool such as the Simplex algorithm. The syllabus normally includes more sophisticated computational techniques, such as the Branch-and-Bound procedure, the Lagrangian Relaxation or the Gomory Cuts, and, recently, , heuristic methods, including problem-oriented heuristics and meta-heuristics (Michalewitz and Fogel 2004).

With respect to the teaching process, OR is no exception to the general rule which establishes that University education, especially in Europe, is still dominated by the objectivist model of learning, i.e., the professor conveys knowledge to the students and relies on each student to master this knowledge (Reiners and Voß 2004, Arbauch and Benbunan-Fich 2006). Implementation of OR techniques also requires some code-writing skills, since the complexity of the problems and the algorithmic nature of the methods usually calls for a computational approach. This is why the teaching of OR, at the undergraduate and graduate levels, has traditionally been carried out from a theoretical perspective, describing the function of the procedures with little more than simple practical illustrations. As a result, and given the complex mathematical nature of the contents, students face the subject with a sense of intimidation (Cochran 2009). A medium-term consequence of this approach is the rejection these students feel towards the OR field when they leave the University and become planners or managers, despite the benefits these quantitative tools may provide in their jobs (List 2004).

Nevertheless, the twenty-first century has seen the introduction of innovative OR teaching concepts and other mathematical concepts, as opposed to the traditional lecturing-and-examination procedure (Burton and Haines 1997). Beginning in the 1990s, a paradigm shift in OR pedagogy began to emerge. Undergraduate classroom discussions began to more substantively address model building, interpretation of results, and implementation issues than the mathematics involved (Cochran 2009, Miranda and Nagy 2011). The new pedagogical approaches were accompanied by the introduction of case-based (Bell and Haehling Von Lanzenauer 2000, Cochran 2000), problem-based (Goodnough 2006, Perrenet et al. 2000) or even project-based (Armacost and Lowe 2003) methodologies.

Many examples in the literature demonstrate the benefits of using specifically designed software for teaching OR techniques, with built-in algorithms for a series of predefined problems (Moore 2001, Geiger 2006, Baloukas et al. 2009, Jing and Zhaotong 2010, Hill and Lam 2014, Kress and Dornseifer 2015). Lourenço (2005) stresses the importance of focusing on modelling and generic concepts rather than algorithm implementation when teaching metaheuristics to Business School students with limited coding skills. Reiners and Voß (2004), who also use an educational software package in their metaheuristics classes, state that "even though the learner has to provide some code fragments . . . , it is almost impossible to ask for a full understanding of the framework and its implementation." Bütün (2005, p. 223) requires students to implement algorithms in a course focused only on genetic algorithms, although this implies group assignments and division of work, previous knowledge of the engineering problem to be solved, and use of out-of-class time for benchmark and progress meetings.

Our objective, as described in the following sections, is the individual implementation of heuristic and metaheuristic methods by students, seeking outcomes such as those presented by Moura Oliveira (2005) in a paper purportedly built by comparing the results obtained by students after applying numerical algorithms to an automated control problem, but where no educational details are provided. In the remainderof this paper, §2 describes the configuration of the "Optimization" postgraduate course at the Seville Engineering School; §3 describes the results of this teaching Methodology; and §4 elaborates on the lessons learned from the experience. Section 5 summarizes this paper..

2. Course Design and Methodology

The "Optimization" course is part of the second semester of the master's degree in Industrial Engineering and Management, offered in the School of Engineering at the University of Seville. It is a research-oriented degree program (as opposed to other industry-oriented master's programs taught at the same institution). Approximately 10–15 students per year take this course. Roughly half of them are industrial engineers recently graduated from the same school; the other half is composed of professionals with 5–10 years of experience in the industry or the public sector. This master's degree constitutes a compulsory stage for students seeking to complete their doctorates in industrial engineering at the University of Seville; approximately 25% of the students enroll in the master's course as part of their doctorate scholarship.

2.1. The Students' Background

A typical distribution of classroom students would be as follows: (a) between 1 and 3 doctorate grant holders (students who have a scholarship to complete their doctorate), (b) between two and four young engineers with a research contract but without a doctorate orientation, (c) three or four vocational future doctoral candidates (who are seeking to obtain a doctorate but who do not have any financial support), and (d) four or five long-time engineers from the industry or the public sector. The coding skills vary significantly from one group to another, usually very good in (a), acceptable in (b) and (c), and rather poor in (d).

Unlike other master's degrees in management taught at economics faculties, our curriculum has a strong bias towards quantitative approaches, with courses such as "Simulation," "Decision-making tools," and "Production Management" running in parallel. The second-semester "Optimization" course follows a first-semester course on "Logistics and Distribution," in which students learn to describe, interpret, and formulate Mixed-integer Linear Programming (MILP) models for complex logistics problems. Before taking the "Logistics and Distribution" course, students in groups (a) and (b) were already familiar with mathematical programming models from an undergraduate OR course, whereas those in groups (c) and (d) typically had no previous OR knowledge.

2.2. The Structure of the Course

Once the students have seen models and become familiar with their uses,, it is time to learn to use models to solve specific problems. This is where the course in "Optimization" fits in. The course runs through the whole second semester, on a weekly basis, two hours per week. It follows the plan shown in Table 1, which covers the three main sections of the course: exact methods, commercial software packages, and heuristic methods.

(a) Optimal solutions: The first section of the course, covering weeks 2 and 3, focuses on exact solution approaches, including contents on the Branch-and-Bound and Branch-and-Cut procedures, the Lagrangian Relaxation, and the Gomory Cuts. Given the time restriction, the procedures are described briefly, simply

Table 1 Course Plan

Week	Contents
1	Presentation and introduction of contents and course requirements
2	Exact methods: Branch-and-Bound, Branch-and-Cut
3	Exact methods: Lagrangian relaxation, Gomory cuts
4	Commercial software packages. Modelling of problem.
5	Description of Metaheuristics
6	Description of Metaheuristics
7	Discussion of solutions obtained with commercial software packages
8	Brainstorming and problem-solving
9	Brainstorming and problem-solving
10	Problem-solving
11	Problem-solving
12	Problem-solving
13	Presentation of results and discussion
14	Presentation of results and discussion

to give students an insight on what exact optimization consists of and the computational difficulties involved. This first section is taught on a theoretical basis, to serve as an introduction, and illustrated with trivial size examples.

(b) Commercial software packages: The second section focuses on the use of commercial software packages to solve linear optimization problems. Practical student participation begins here. The students are presented with several free and licensed packages and a small problem (the same one for all students). They are asked to build the linear formulation for that problem, feed it to one of the packages, and submit and analyze the solution. They are free to choose the package they want to use, normally aiming for spreadsheet applications, specific packages such as Lingo® or Gurobi® or general purpose systems such as Matlab®. The process of submission and discussion of results explains why this second section is split between weeks 4 and 7, i.e., to give students enough time to complete their assignments.

(c) Heuristic methods: The third and final section starts in weeks 5 and 6 with a brief review of the basic principles of the different types of meta-heuristics and other heuristics applicable to the type of problem students must solve. At the end of the sixth session, students are presented with a bundle of folded pieces of paper, each containing the name of one of these heuristic procedures, and they are asked to pick one. That same day, through the University Web interface, they are provided with data for the same optimization problem they had to solve with the commercial package. This time, however, the problem is much larger so that it is impossible to find the optimal solution using the same software in a reasonable amount of time. Again, the problem to solve and the subsequent data is the same for all the students, and all students are required to apply their selected procedures to solve it. Table 2 lists the methods applied to the different problems proposed thus far in the "Optimization" course.

2.3. The Teaching Approach

O'Brien et al. (2011) confirm that students learn tools best by trying to use them, and that the best way to induce that use is through practice-based learning, group-based learning, and reflective practice. These are the key elements that we have tried to bring into the last part of our heuristics course, keeping in mind that the previous work and the skills, attitudes, and dispositions developed by the teacher are essential to address the complexities involved in this type of procedure (Goodnough 2006).

After the draw, students have two weeks to reflect on the problem, analyze the data, and find and read references on the subject. (Before handing out the assignments, we surfed the Web to make sure that there are scientific publications describing applications to the selected problem using each one of the proposed methods. Fortunately, the OR-related papers of this type are numerous in the literature .) Weeks 8 and 9 are brainstorming and problem-solving sessions, where the students, under the supervision of and with help from the lecturer, find a solution to the first questions that are common to all, i.e., how to encode the problem, how to evaluate solutions, how to generate an initial solution, how to make the algorithm stop, etc.

Table 2 **List of the Heuristic Methods Applied in the Course**

Generic heuristics	GRASP divide and conquer
Metaheuristics	Simulated annealing Genetic algorithms Tabu search Ant colony optimization Particle swarm
Problem-specific heuristics	
Container loading (Chen et al. 1995)	Heuristics A, B1 and B2 in Bischoff and Marriott (1990)
Uncapacitated facility location problem (Klose and Drexl 2005)	Steepest descent heuristic in Ghosh (2003)
p-Hub median problem (Skorin-Kapov et al. 1996)	Single-exchange, double-exchange and clustering heuristics in Klincewicz (1991)
Multiple knapsack problem (Martello and Toth 1987)	Heuristics I1 and I2 in Martello and Toth (1981)

The degree of autonomy in their work varies among students, particularly depending on their computer coding skills: Those who are familiar with algorithms and routines quickly grasp the basics of heuristic procedures, although those who have never before faced an algorithm can still sometimes give the correct answer to specific questions. This is particularly rewarding. The teaching methodology of these sessions starts with analysis of an optimization problem (a traveling salesman problem (TSP)) different from the one they will be asked to solve. The teacher proposes and explains the benefits of different encodings (usually the most difficult concept for those students who are unfamiliar with computer science), focusing only on basic vector and matrix data structures. The students with better coding skills immediately identify object-oriented approaches. From there on, evaluation and generation of initial solutions (they are simply asked for straightforward random procedures) are easy for the students to accomplish. The requested termination criterion depends only on the number of iterations.

These collaborative problem-solving sessions are essential to engage the students in course contents and to develop their understanding of the heuristics' operation (Nordstrom and Korpelainen 2011, Garcia-Perez and Ayres 2012). From there on, the students are more or less left on their own to find solutions to build their own code, depending on their specific heuristic, supported by the instructor through the three following sessions, in weeks 10, 11, and 12. The main difficulties at this stage are usually coding-related, but some help on parameter calibration or data interpretation is often required. Help is usually needed with the concepts not directly related to the heuristic algorithms themselves (e.g., the concept of local optima or understanding why the existence of many identical chromosomes in the population of a genetic algorithm does not directly imply that the algorithm has converged to the optimum). This is where the greatest efforts in comprehension will be needed. As with commercial software, the students are free to choose the implementation language they wish. Because it is an ad-hoc implementation, we suggest they start by applying the heuristic method to the small-sized problem (the one they already solved with the commercial package, and whose optimal solution they already know). Once that is working, they can move on to the large-sized problem.

The course closes with two sessions where the students present their results to the whole class, paying special attention to the following aspects: a brief outline of their heuristic method, and how they implemented it for the problem in question; the final solution they found, and the evaluation result for that solution; and the amount of computational time required by their algorithm to find that solution. After the presentations, the lecturer begins a discussion of the different solutions found by the different heuristics, the fact that some methods work better than others, and the different amounts of time consumed by each one. Thus, the three conditions of effective learning, as declared by Alavi et al. (1995) are fulfilled i.e., active learning by doing, cooperation and teamwork in learning, and learning through problem solving.

2.4. Addressing Coding Problems

One issue the students raise when introduced to the course methodology in session 1 is the need for software coding. Implementation of algorithmic techniques and their application to the specific problem assigned every year requires coding. Although some of the master's students are comfortable with this, others feel they are incapable of coding a heuristic algorithm. Because the assignments and student evaluation criteria must be common, the less skilled students usually set the pace for the class. We considered setting different "levels" in the class depending on the student coding skills, but discarded this option as unfair in terms of student evaluation. (Note that, when assigning grades at the end of the semester, we do keep in mind the background and effort invested by different students).

Our response to coding apprehension is always the same: The objective of this course is to evaluate the students' understanding of heuristic methods, not their coding skills. One of the reasons we devised this course in this way is our feeling that students engaged in a quantitative doctorate-oriented master's program should have coding skills, at least at the undergraduate engineering level. Given that basic heuristic methods are not complex mechanisms, this course provides a good opportunity to acquire or renew those skills. oweve Nevertheless, for those students who do not feel they can master software coding, we provide an alternative: Those students are allowed to perform the calculations for a heuristic using spreadsheet software, with the calculations for each iteration of the heuristic on a separate worksheet. Although this cannot replicate the performance of an automated heuristic procedure, students can easily perform several dozen iterations of their heuristic, thereby improving the initial solution and demonstrating understanding of how the method works.

3. Achieved Outcomes

Table 3 shows the optimization problems that have been assigned for students to solve over the four years the "Optimization" course has been offered. The small-sized problems were solved using commercial solvers and the large-sized problems with heuristic methods. The size of each problem is defined by one or more specific parameters. For example, the size of a container loading problem is defined by the number of boxes available. Given the value of the size parameter, the mathematical model for the problem results in a number of variables (also in Table 3), which determine whether the problem can be solved with a commercial package (if the number of variables is not too large). Data for the small and large sizes of the four problems were taken from the well known OR Library (Beasley 1990).

Next we describe the students' perception of this new methodology as they worked on those optimization problems over the years, including how students were evaluated, how they perceived the collaborative aspects of their work, and the feedback they provided.

3.1. Student Evaluation

Heuristic methods are optimization procedures that often incorporate a random component (especially in the case of metaheuristics). Not all of them work equally well when applied to specific optimization problems. Thus, evaluation of students in this course is not related to the quality of the solution their assigned method finds for the problem or to how fast that solution is found. Grades are related to the students' ability to implement and present their work, to explain the procedure, and to discuss the results in class. It is not a competition to find out who obtains the best solution to the problem, but rather an inclusive approach, using comparison between the different results to validate those results: Students soon learn that obtaining too good or too bad results with respect to the

Table 3 Description and Sizes of the OR Problems Used so Far in the "Optimization" Course

		Value of the size parameter		Resulting number of variables	
Problem	Size parameter	Small size	Large size	Small size	Large size
Container loading	No of boxes	5	112	220	88,816
Uncapacitated facility location problem	No of potential warehouses and no of customers	3 and 10	50 and 50	40	2,550
p-Hub median problem	No of customers	5	25	130	15,650
Multiple knapsack problem	No of knapsacks and no of products	10 and 28	30 and 500	280	15,000

rest of the class typically means that they got it wrong. Furthermore, students know from the beginning that those choosing the spreadsheet option to implement their methods instead of writing code are likely to receive lower grades, since they would be unable to participate in the validation-through-comparison process.

3.2. Student Collaboration

Even though they are not required to do so, students tend to apply a collaborative approach with respect to the common features of heuristic methods. Whether splitting up the work or completing it together, they all benefit from the fact that all their assignments should include the same coding, generation and evaluation of solutions, and termination conditions. This "collaboration without replication" methodology is the key to the teaching process. Some parts of the assignments are identical, and therefore the work can be shared, but in the end the assignments are all different, and each one corresponds to a different method.

We need to note here that in general, Spanish teachers, when applying teaching techniques that are not based on the traditional study-and-exam process, are not always keen to promote collaborative work between students fearing that not all students will show the same degree of involvement (**?**Clariana et al. 2013). Although students in the "Optimization" course appear shocked by the possibility of working together "legally" on individual assignments, at the same tim they realize that they must work individually on the specific components of their own heuristic method. Attempts to incorporate this methodology in other courses have been difficult, as it forces the teacher to prepare a different assignment for each student. Here, however, the same assignment is solved by each student with a different technique, which greatly facilitates course preparation.

3.3. Student Feedback

Students' opinions on course planning are unanimously positive. The most common feedback includes expressions such as "challenging," "tough but rewarding," and even "fun." This feedback is unusual for OR teachers. These opinions also have an anonymous quantitative component: They are expressed through the Student Survey Program of the University of Seville, which administers a survey to all students attending each University course. The results obtained by the "Optimization" course over the last three years and their comparison with the master's, Department, and University averages are shown in Figure 1. Despite the complexity of the preparation and the teaching process, the positive consistency of these results has convinced us to maintain what was originally a pilot program.

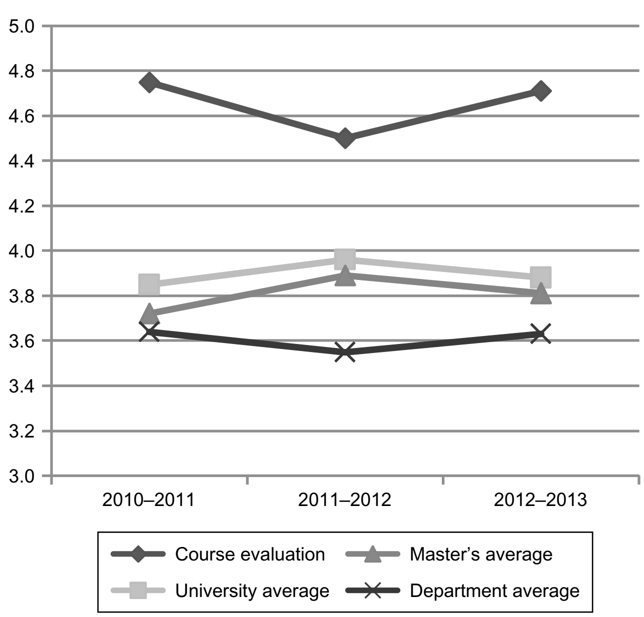
4. What the Teachers Learned

As mentioned earlier, we implemented this teaching technique as an experiment. We have learned a great deal since the course began. This section, mainly composed of our personal opinions, compiles these findings, which might be useful to other teachers who are considering applying this problem-based approach in their OR classes.

4.1. Providing Conceptual Insight

The first difficulty in the course is the actual understanding of what the student is trying to solve. Rather

Figure 1 Results of the student feedback survey for the Optimization course



than counting on students to have a deep insight into the type of problems they are facing (Bütün 2005), this course is the first opportunity for our students to learn about combinatorial-type problems. Although they know about logistics- or production-related optimization problems and mathematical formulation, the leap to an algorithmic approach to the problem is not always straightforward. Thus, our first objective must be to introduce them to the characteristics of these problems. This challenge was usually related to the encoding of solutions and understanding of the fitness concept.

With respect to encoding, the instructor began with the idea of using a vector (with continuous or binary entries). Students quickly suggested correct options for the TSP and the knapsack problem. However, the more complicated problems, such as the vehicle routing problem (VRP) or the container loading problem, required more intervention in the discussions. We felt that taking our technique as far as letting the students implement any encoding they could think of to see whether it was correct would require too much time and student effort.

Moreover, the difficulty in understanding evaluation of the fitness value also depended on the actual problem faced by the students. While routing or knapsack problems were no obstacle, location problems were more challenging. Long discussions were required to understand that allocation was straightforward (and therefore did not need to be incorporated in the encoding of the solution) once the location was defined. Another difficult debate arose from the misunderstanding of the *minimax* approach in location problems (Goldman 1972) as an entirely different solution technique, instead of simply a different way of formulating the fitness calculation.

4.2. The Coding Issue

In §2.4 we described how we offer students with uncertain coding skills the alternative of performing their calculations in a spreadsheet; in §3.1 we discuss how that alternative would be evaluated. When we began offering this choice , we expected a significant percentage of our students (particularly those referred to in §2.1 as the "*d*" group) to take it. Although concerns about having to write code have been raised every year, thus far not a single student has chosen the spreadsheet alternative. Students with higher coding skills are happy to actually "solve" a problem. Those with lesser skills, despite delivering more rudimentary code lines, usually end the course with a feeling of achievement. In short, they worked hard but they made it, and at least they did not have to pass an examination.

After encoding and fitness calculation issues have been resolved, the most common requests for help are related to the fact that the algorithm "does not work properly." Here the instructors can only provide guidelines on how to carry out partial assessments of different parts of the algorithm to locate the problem. In the end, those students who are more familiar with coding are usually impressed by the simplicity of the algorithms. This led us to consider creating work teams composed of students with different skills and backgrounds. We decided against this idea for two reasons. First, we felt it could cause students to concentrate on only part of the process. Second, we felt that some team members would then be unable to start working until others had finished.

4.3. Knowledge Applicability

Expecting this course to equip all students with sufficient skills to apply OR techniques in their careers is, in our opinion, overly optimistic. Here too, the applicability of the acquired knowledge depends very much on the groups in which our students fall. . On the one hand, the applicability is immediate for doctoral students and their research work. Eight of those who took this course in the first four years have finished or are about to finish their theses; they have all used meta-heuristics, and this course was their first contact with these optimization techniques. This is perfectly in line with the main objective of the course, given the doctoral orientation of this master's degree.

On the other hand, applicability of the knowledge gained in this course is not as direct for those students working in the industry. The leap from standard OR problems used in course assignments to actual production management or logistics problems in their jobs is too large , in terms of defining the scope of the problem, understanding its variables, and defining it mathematically in the form of a fitness function. Simply put, we feel this is asking too much from a course like ours. Nevertheless, these students will always be aware of the existence of heuristic and meta-heuristic algorithms that can be used to solve complex problems, and of how these algorithms work, even if they have to ask for further help when facing a specific industry problem. For example, one of our former students, who is currently working in the aeronautical industry, recently requested our Department's help with the application of heuristic computing methods to optimize a particularly complex logistics process.

4.4. What the Course Achieves and Lacks

The objective of our course is to teach students how heuristics and meta-heuristics work. We believe this objective is achieved by our problem-based approach. The old theoretical approach involved teaching the basics of meta-heuristics on the blackboard, and using the blackboard to provide students with examples of how crossover operators or tabu lists work. In terms of heuristics design and implementation, the

"Optimization" course may be capable of much more, but it is unclear how the content should be delivered given existing constraints on time and student background. Nevertheless, based on what our students were learning and what they are learning now from the actual implementation of heuristic techniques, we feel that the success of our new approach is quite evident.

Nevertheless, after four years using our approach we feel that one primary limitation is that our course lacks perspective. Even though there are few publications with enough detail on real life applications in heuristics and meta-heuristics, and which would be appropriate for a master's course (Lourenço 2005), given our deep focus on algorithm implementation, showing students what these techniques are capable of, how they can be used, and how to formulate optimization problems based on industry challenges may well be overlooked.

4.5. Opportunities for Improvement

Based on our experience, the primary trade-off to be addressed in planning our course for the future is between exact methods and heuristics. In the original course design, our academic backgrounds propelled us toward teaching exact approaches before moving on to heuristic techniques. It may be that this consumes too much time, which could be used differently. The new graduate course' orientation toward practical contents will mean leaving aside the theoretical foundations of exact mathematical techniques to concentrate on the use of commercial solvers and their applicability in real-life industrial environments.

The time saved could well be used to resolve the other trade-off that we find in our course, i.e., between knowing how to implement and knowing how to calibrate, analyze, and refine heuristic procedures. The calibration, analysis, and refinement aspect is mentioned only briefly in our current course contents. Apart from using some time to discuss actual industrial applications in the literature, asking students to perform calibration runs, to improve the solutions provided by their algorithms or to try to reduce computational times, would lead them further towards full applicability of heuristic techniques.

Finally, on a more practical note, we have found that allowing our students to plan their own work might not be the best option. Many implementation problems come from students who have drifted through (or even missed) problem-solving sessions, and who then encounter countless difficulties as the submission deadline approaches. Thus, our future course plan should include partial submissions (for example, submit the fitness function first, then the neighborhood search, then the iterative routine, etc.). This should force students to maintain a steady rhythm of work during the semester.

5. Conclusions

Operations research techniques and the application of heuristic techniques are challenging for engineering students to learn because of the mathematical and computational aspects. We have presented the teaching methodology developed in our postgraduate optimization course, consisting of engaging students in the actual implementation of heuristic techniques to solve a specific problem. We incorporated this methodology as a pilot test four years ago, expecting a near mutiny scenario from students who, until then, just listened to lectures, memorized concepts, and took exams. Although we were prepared to abandon the challenge as too demanding, the optimal results from the first year have continued, and the course structure is now consolidated. Students have expressed positive opinions after submitting their assignments and receiving their grades. They feel they have learned something they are unlikely to forget in the short term, something that will remain with them as an additional tool to complete their doctoral studies and help them in future research.

The application of this problem-based approach was new to our students. This newness contributed greatly to the success of the course, as did the feeling, not incited by their teachers, that they were "competing" to obtain the best algorithm performance. We have benefited from having a small number of students in the classroom, although the amount of time required to help solve coding issues has so far prevented us from devoting more time to more advanced aspects of OR, such as calibration, algorithm performance or more sophisticated stopping criteria. Introduction of these aspects to the course is one of our short-term teaching objectives.

Of course, we accept the criticism of allowing our students learn only one heuristic method, i.e., the one they are applying and implementing. Nevertheless, the characteristics of the course enable us to proceed in this way. When the students attend the two lectures on algorithm architecture (weeks 5 and 6) they know they will have to apply one of those techniques to the given problem but they do not yet know which one. The intensity of their attention and participation in those two lectures is something we do not find in any other course we teach. Apart from this, the primary aspect of the course is learning the philosophy of the method, rather than a specific heuristic technique. We know from our own research experience that after implementing a specific algorithm to solve an optimization problem it becomes easy to implement any other algorithm. In short, learned one, learned all (if you really learned it).

Acknowledgments

The authors wish to express their gratitude for the comments and suggestions raised by the editor, associate editor, and reviewers, which helped to improve it significantly with respect to the initial version. The authors also wish to acknowledge the support provided by the University of Seville through its 2015 Support Programme for Teaching Innovation and Improvement.

References

Alavi M, Wheeler BC, Valavich JS (1995) Using IT to re-engineer business education: An exploratory investigation of collaborative telelearning. *Management Information Systems Quart.* 19: 293–312.

Arbauch JB, Benbunan-Fich R (2006) An investigation of epistemological and social dimensions of teaching in online learning environments. *Acad. Management Learn. Ed.* 5(4):435–447.

Armacost AP, Lowe JK (2003) Operations research capstone course: A project-based process of discovery and application. *INFORMS Trans. Ed.* 3(2):1–25.

Baloukas T, Paparrizos K, Sifaleras A (2009) An animated demonstration of the uncapacitated network simplex algorithm. *INFORMS Trans. Ed.* 10(1):34–40.

Beasley JE (1990) OR-library: Distributing test problems by electronic mail. *J. Oper. Res. Soc.* 41(11):1069–1072.

Bell PC, Haehling Von Lanzenauer C (2000) Teaching objectives: The value of using cases in teaching operational research. *J. Oper. Res. Soc.* 51:1367–1377.

Bischoff EE, Marriott D (1990) A comparative evaluation of heuristics for container loading. *Eur. J. Oper. Res.* 44:267–276.

Burton L, Haines C (1997) Innovation in teaching and assessing mathematics at university level. *Teaching in Higher Ed.* 2(3): 273–293.

Bütün E (2005) Teaching genetic algorithms in electrical engineering education: A problem-based learning approach. *Internat. J. Electrical Engrg. Ed.* 42(3):223–233.

Chen CS, Lee SM, Shen QS (1995) An analytical model for the container loading problem. *Eur. J. Oper. Res.* 80:68–76.

Clariana M, Badia M, Cladellas R, Gotzens C (2013) Academic cheating and gender differences in Barcelona (Spain). *Summa Psicológica UST* 10(1):65–72.

Cochran JJ (2000) Successful use of cases in introductory undergraduate business college operations research courses. *J. Oper. Res. Soc.* 51(12):1378–1385.

Cochran JJ (2009) Pedagogy in operations research: Where has the discipline been, where is it now, and where should it go? *Orion* 25(2):161–184.

Garcia-Perez A, Ayres R (2012) Modelling research: A collaborative approach to helping Ph.D. students develop higher-level research skills. *Eur. J. Engrg. Ed.* 37(3):297–306.

Geiger MJ (2006) Teaching modern heuristics in combinatorial optimization. Kumar D, Turner J, eds. *Internat. Federation for Inform. Processing, Volume 210, Ed, 21st Century—Impact of ICT and Digital Resources* (Springer, Boston), 65–74.

Ghosh D (2003) Neighborhood search heuristics for the uncapacitated facility location problem. *Eur. J. Oper. Res.* 150(1): 150–162.

Goldman AJ (1972) Minimax location of a facility in a network. *Transportation Sci.* 6(4):407–418.

Goodnough K (2006) Enhancing pedagogical content knowledge through self-study: An exploration of problem-based learning. *Teaching in Higher Ed.* 11(3):301–318.

Hill SE, Lam M (2014) A teaching exercise for the travelling salesman problem with time windows using real-world data. *Internat. J. Inform. Oper. Management Ed.* 5(4):363–375.

Hillier FS, Lieberman GJ (2001) *Introduction to Operations Research* (McGraw Hill, New York).

Jing L, Zhaotong Z (2010) A learning tool of genetic algorithm. *ETCS 2010, Internat. Workshop on Ed. Tech. Comput. Sci.*, 443–446.

Klincewicz JG (1991) Heuristics for the *p*-hub location problem. *Eur. J. Oper. Res.* 53:25–37.

Klose A, Drexl A (2005) Facility location models for distribution system design. *Eur. J. Oper. Res.* 162:4–29.

Kress D, Dornseifer J (2015) LAVES: An extensible visualization tool to facilitate the process of learning and teaching algorithms. *INFORMS Trans. Ed.* 15(3):201–214.

List B (2004) Progress report on the INFORMS initiative to market the profession. *ORMS Today* 31(1):22–23.

Lourenço HRD (2005) Teaching metaheuristics in business schools. Economics and Business Working Paper 806, Universitat Pompeu-Fabra.

Martello S, Toth P (1981) Heuristic algorithms for the multiple knapsack problem. *Computing* 27:93–112.

Martello S, Toth P (1987) Algorithms for knapsack problems. *Ann. Discrete Math.* 31:213–258.

Michalewitz Z, Fogel DB (2004) *How to Solve It: Modern Heuristics* (Springer, Berlin).

Miranda J, Nagy M (2011) A case of cooperation in the European OR education. *Eur. J. Eng. Ed.* 36(6):571–583.

Moore M (2001) Teaching students to use genetic algorithms to solve optimization problems. *J. Comput. Sci. Colleges* 16(3):19–25.

Moura Oliveira PB (2005) Modern heuristics review for PID control systems optimization: A teaching experiment. *Internat. Conf. Control and Automation (ICCA2005), Budapest*.

Nordstrom K, Korpelainen P (2011) Creativity and inspiration for problem solving in engineering education. *Teaching in Higher Ed.* 16(4):439–450.

O'Brien FA, Dyson RG, Kunc M (2011) Teaching operational research and strategy at warwick business school. *Informs* 12(1):4–19.

Perrenet JC, Bouhuijs PAJ, and Smits JGMM (2000) The suitability of problem-based learning for engineering education: Theory and practice. *Teaching in Higher Ed.* 5(3):345–358.

Reiners T, Voß S (2004) Teaching meta-heuristics within virtual learning environments. *Internat. Trans. Oper. Res.* 11:225–238.

Skorin-Kapov D, Skorin-Kapov J, O'Kelly M (1996) Tight linear programming relaxations of uncapacitated *p*-hub median problems. *Eur. J. Oper. Res.* 94:582–593.

Strayer JK (1989) *Linear Programming and Its Applications* (Springer, Berlin).

Winston WL, Venkataramanan M, Goldberg JB (2003) *Introduction to Mathematical Programming*, Vol. 1 (Thomson/Brooks/Cole, Pacific Grove, CA).