Student's Approach to Linear Programming Modeling

Lioubov Dombrovskaia[1] and Liliana Guzmán[2]
1 Departamento de Informatica, Universidad Tecnica Federico Santa Maria, av. Espana 1680, Valparaiso, Chile[7]
liuba@inf.utfsm.cl
2 Departamento de Computacion, Universidad de Valparaiso, av. Gran Bretana 1091, Playa Ancha, Valparaíso liliana.guzman@uv.cl

Abstract. Software design implies searching for and establishing an adequate morphism between the real world and the desired software. Morphisms establish correspondences between different domains while some properties are preserved, at the same time. It allows seeing different things as the same, taking the substitute image for the real one. The more adjusted to reality the morphism is, the better the system models the real situation. We propose the use of morphisms as a pedagogical tool in order to teach object-oriented concepts and also to promote better software design. We developed a course based on the explicit use of morphisms. Through experimentation, we compared the results with an equivalent course not using morphisms. From the results we may infer that using morphisms helps to develop strategies to analyze and to construct adequate software models.

1 Introduction

This study aims to understand how students acquire the ability to model linear programming (LP) problems. Our motivation is to improve LP teaching through the employment of better teaching methods and modeling languages.

Current studies on teaching methods denote a dissatisfaction with traditional operations research (OR) courses and propose multiple changes [9,10]. Murphy and Pachandam [7] studied experimentally the impact of teaching methods on the student's acquisition of modeling skills. Their schema earlier approach provides a framework to organize knowledge based on analogies and what is demonstrated to

[7] The authors are grateful to the Chilean National Science Foundation (FONDECYT) for its partial support in this research under grant N°1010125.

be the most effective method. Stevens and Palocsay [12] studied the difficulty of verbal problem solving and demonstrated that a translation approach consisting of successive phrase reformulations to formulate verbally a constraint improves the success of LP modeling.

Modeling languages have evolved through three stages: solver-oriented (such as MPS), analyst-oriented (such as AMPL), and visual (such as MGPL) [5]. Algebraic languages such as AMPL [2] are the most popular today. LP languages can be essentially different from the point of view of the quantity of work, ability and understanding required from the LP analyst [4]. Two criteria have been proposed to measure the analyst's effort: Murphy *et al.* [6] define *work intensiveness* as the amount of detailed work required from the analyst, and Geoffrion [3] declares *ease of use* as a required property of modeling tools.

The need for testing potential languages has been stated by several authors for a long time [5,7,8,11], but no such studies have been published to the best of our knowledge. The available knowledge about the modeling languages and teaching process is scarce in the area of LP modeling. Therefore, the goal of the quasi-experiment herein described is to test the criteria for measuring the effectiveness of a modeling process. We intentionally avoid experimental jargon to make this research more understandable to the readers.

Section 2 describes the results of evaluating the AMPL language, discussing briefly issues of modeling efficiency and style. The article finishes with conclusions and future work.

2 Experimental study

A characterization of the modeling process should describe how an analyst models a problem using a specific language. Comparative characterizations can improve the teaching process, and/or give feedback on how the language should be. Thus, the experimental study should respond to the following questions: How do participants perform and what are their major difficulties? How do participants model?

2.1 Did students learn LP modeling?

Fifty-six computer science students participated in the experiment, 50 male and six female, with an average age of 21 years. Their average cumulative grade has a mean of 59%. None of the students had previous LP training. The experiment began with training and continued with performance measurement in laboratory sessions. The training took four sessions. The teaching method was based on schema formation and analogical reasoning that is shown to be the most effective [7]. The most common problem types are included in the training: production, blending, transportation, assignment, and fixed costs treatment [2,13].

Training effectiveness was measured at the last session through a test that asks participants to identify the correct formulation of model parts in AMPL. Most students recognized the correct model but picked a syntactically wrong version over

a syntactically correct one. It was expected that this problem would be overcome when working in front of a computer.

Afterwards, participants had 6 laboratory sessions. The first session introduced them to the use of a modeling environment, and the following ones measured their individual modeling performance for increasingly complex problems. Students did not receive any feedback on their work, so they developed their own quality control methods. The modeled problems belong to problem types covered in training and are adapted from well known textbooks [2,13] to be solvable in less than 1.5 hrs. The modeled problems will be hereafter be called "swimmers", "mining", "generators" (electrical), "forest" and "warehouse" and were presented in the same order as named here. Modeling performance is characterized by the following two aspects:

- **Quality**: closeness of the obtained model to the optimal model (rank: 0-100).
- **Solution**: indication of the fact that a solution can or can not be obtained from the model (rank: 0 or 1).

Students' performance in each laboratory varies from very bad to excellent (Table 1), but taking into account the increasing problem complexity, it can be said that students really learned LP modeling. Most participants developed acceptable models, but less than half of the students reached any solution except in one problem ("swimmers"), where 50% of the participants found a solution. The model quality is similar throughout all the laboratory sessions: there is no significant quality difference between them except for the "forest" problem, whose significantly lower model quality is probably due to its difficulty.

Table 2. Quality and solution rate

Variable	Swimmers	Mining	Generators	Forest	Warehouse
Quality mean (σ)	62.7 (28.2)	58.9 (16.8)	55.8 (15.1)	45.34 (8.1)	56.2 (24.8)
Solution (%)	50	33.9	30.4	21.4	30.4

Students had more difficulties defining constraints, then variables and objective functions. In fact, the difference between the students who reach a solution and those who could merely develop an acceptable model is explained by the ability to recognize those model components. This result is noteworthy because the problem statements actually make the component identification easy according to the instructors. Therefore, it seems that a solid algebraic background is necessary but not sufficient for effective learning of LP modeling.

2.2 How did students model?

The modeling style was studied by considering the number of iterations and executions and their relation with quality and solution. These two numbers were collected through a log file recorded for each laboratory session of each student.

- **Iterations**: number of model trials whether or not resulting in errors.
- **Executions**: number of model executions leading to a solution of the problem.

Students who reached a solution presented a high number of executions for all problems. The number of executions was alike across all laboratory sessions with the exception of the "forest" problem (Table 2). This difference can be explained by the comparatively bigger data volume of the "forest" problem. However, some students never passed the syntax revisions, which means that they never got to execute their models.

Table 3. Number of iterations and executions

Variable	Swimmers Mean (σ)	Mining Mean (σ)	Generators Mean (σ)	Forest Mean (σ)	Warehouse Mean (σ)
Iterations	19.7 (11.6)	17.2 (13.9)	17.0 (17.1)	8.7 (11.3)	12.2 (10.7)
Executions	6.1 (4.8)	5.2 (6.9)	6.9 (11.9)	3.3 (6.2)	3.9 (5.6)

Table 2 also indicates that students sustained a high number of iterations throughout the laboratory sessions. All students had syntax problems, but only some of them corrected their errors through trial-and-error; all other students were unable to solve the syntax problems or simply did not know how to model.

The higher the number of executions, the higher is the quality of the model and the achievement of the solution. Therefore, some students develop an iterative style that allows them to improve their models and reach a solution.

3 Conclusions

The presented experiment allows measuring different aspects of the modeling process. Its most important findings are:

- Most students develop acceptable models, but less than half of them reach any solution due to syntax and semantics errors.
- Syntax errors are not a minor issue in AMPL; students do not master the syntax.
- Students always use the trial-and-error approach to solve syntax and semantics errors, instead of using the syntax guide.
- Modeling is difficult for the students.

The teaching of LP modeling with algebraic languages can be improved by the findings of this experiment. The reinforcement of schema formation should improve the modeling skills and diminish the number of semantics errors, while the reinforcement of syntax or the creation of languages with better syntax should diminish the number of iterations. Both aspects lead to time savings and therefore should improve the models' quality.

This research established and tested objective factors to evaluate the effort required from LP analysts: model quality, solution, number of iterations and number of executions. The experiment findings are consistent with existing studies [7,12]. Clearly, the method needs to be tested with other languages, and ongoing work is evaluating a visual LP language [1] with the same treatment, enabling later comparison between both languages. Of course, the answer to the question of which

language is better suited for an LP analyst requires further testing with other languages and other audiences.

References

1. Dombrovskaia L., Rodriguez P., Nussbaum M. (1998) Knowledge based modeling tool for linear programming. Comput Oper Res 25: 379-388
2. Fourer R. M.Gay D., Kernighan B. (1993) AMPL. A modelling language for mathematical programming. First Edition. The Scientific Press Series
3. Geoffrion H. I. (1987) Introduction to structured modelling. Manage Sci 33: 547-588
4. Greenberg H., Murphy F.H. (1995) Views of mathematical programming models and their instances. Decis Support Syst 13: 3-34
5. Jones C.V. (1994) Visualization and optimization. INFORMS J Comput 6: 221-257
6. Murphy F.H., Sthor E.A., Asthana A. (1992) Representation schemes for linear programming models. Manage Sci 38: 964-991
7. Murphy F.H., Panchanadam V. (1998) Using analogical reason and schema formation to improve the success in formulation linear programming models. Oper Res 47: 663-674
8. Petre M. (1995) Why looking isn`t always seeing: Readership skills and graphical programming. Commun ACM 38: 33-43
9. Pidd M. (1999) Just modelling through: a rough guide to modelling. Interfaces 29: 118-132
10. Robinson S., Meadows M., Mingers J., O\'Brien F., Shale E., Stray S. (2003) Teaching OR/MS to MBAs at Warwick Business School: A turnaround story. Interfaces 33: 67-76
11. Srivanasan, A., Te'eni, D. (1995) Modeling as constrained problem solving: an empirical study the data modeling process. Manage Sci 41: 419-434
12. Stevens S.P., Palocsay S.W. (2004) A translation approach to teaching linear programming formulation. Informs Transactions on Education 4 (3)
13. Williams H. (1999) Model building in mathematical programming. 4th edition. John Wiley and Sons Ltda.