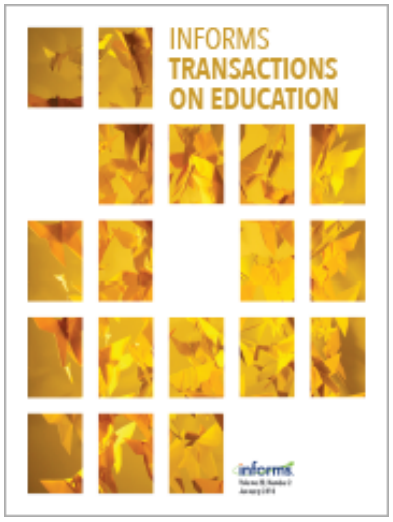

INFORMS Transactions on Education

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

On the Use of Integer Programming versus Evolutionary Solver in Spreadsheet Optimization

Kenneth R. Baker, Jeffrey D. Camm,

To cite this article:
Kenneth R. Baker, Jeffrey D. Camm, (2005) On the Use of Integer Programming versus Evolutionary Solver in Spreadsheet Optimization. INFORMS Transactions on Education 5(3):1-7. https://doi.org/10.1287/ited.5.3.1

Vol. 5, No. 3, May 2005, pp. 1–7
ISSN 1532-0545 | 05 | 0503 | 0001
INFORMS Transactions on Education
informs
DOI 10.1287/ited.5.3.1

On the Use of Integer Programming versus Evolutionary Solver in Spreadsheet Optimization

Kenneth R. Baker
Tuck School of Business, Dartmouth College, Ken.Baker@dartmouth.edu

Jeffrey D. Camm
Department of Quantitative Analysis & Operations Management, College of Business, University of Cincinnati, Jeff.Camm@uc.edu

The introduction of the evolutionary solver in Frontline Systems' Premium Solver for Education allows students to use any functions in Excel for modeling optimization problems. As a result, instructors teaching optimization now face a dilemma of how much emphasis to place on "traditional" integer programming versus the unrestricted but heuristic approach of the evolutionary solver. Our goal in this work is to shed some light on the tradeoffs in these two modeling approaches. We discuss some experimental results comparing the two approaches for a number of well-known problem types. We also report some observations of student performance with these two different approaches.

1. Introduction

The existence of optimization and simulation in spreadsheets has had a dramatic impact on the teaching of operations research (OR) (Erkut 1998). Instead of requiring students to use unfamiliar stand-alone software that most would no longer use after graduation, the spreadsheet allows OR to be taught in an environment students have already accepted and will continue to use. The very fact that OR tools are incorporated into the spreadsheet lends instant credibility to their use. Required courses and electives emphasizing analytical modeling in spreadsheets have gained new-found acceptance and appreciation.

In what ways do spreadsheets change how we teach OR? For one, the spreadsheet environment offers an integrated platform for OR. With relative ease, we can use simulation, optimization, forecasting, data management tools, and graphics in combination because they are all available in the spreadsheet (Camm 1998). Spreadsheet graphics capabilities provide a powerful tool for building intuition in OR and statistics (Camm 2000, Savage 2001). The existence of Visual Basic for Applications in Excel allows for the easy integration of tools as well as the construction of powerful decision support systems (Ragsdale 2001). All these factors have changed how OR is taught.

At present, a number of textbooks come bundled with Frontline Systems' Premium Solver for Education (PSE).[1] The evolutionary solver in this version of the software allows students to use any functions in Excel for modeling optimization problems. Functions such as MAX, IF, INDEX, and COUNTIF, which can derail the linear and nonlinear solvers, are accommodated by the evolutionary solver. Instructors teaching the modeling of combinatorial optimization problems now face a dilemma of how much emphasis to place on "traditional" integer programming versus the unrestricted but heuristic approach of the evolutionary solver. The freedom to choose any function in Excel has implications for how modeling is taught. Consider the popular case of a fixed cost in a production facility. A typical modeling approach using linear integer programming is to define two classes of variables, $x_i =$ amount to produce at facility i and $y_i = 1$ if facility i is open and 0 if it is not open. If C_i is the capacity of facility i, then the linear form $x_i \le C_i y_i$ relates the production quantity and the decision to open the facility. Production can take place ($x_i > 0$) only if the facility is open ($y_i = 1$) and some fixed cost (F_i) is charged to the objective function via a term $F_i y_i$. This same relationship can be modeled for the evolutionary solver without binary variables but with the constraint $x_i \le C_i$ and an objective function component IF($x_i > 0, F_i, 0$).

[1] Frontline's web page lists more than a dozen such textbooks at http://www.solver.com/academic2.htm.

There are tradeoffs in choosing one approach versus the other. Assuming the integer program can be solved in a reasonable amount of time, its solution will be optimal. The evolutionary solver cannot guarantee an optimal solution, but it may often be more intuitive for students. Which approach should be taken in teaching modeling? Our intention in this paper is not to answer that question but to shed some light on the trade-off so that at least more informed teaching may occur. To that end, we discuss in the following section some experimental results comparing the integer programming approach to the evolutionary solver approach for solving four well-known problem types. Then, we report some results on student performance with these two different approaches. In our final section, we provide a summary.

Frontline's web page lists more than a dozen such textbooks at http://www.solver.com/academic2.htm.

2. Computational Study

We compared the performance of (linear) integer programming with an evolutionary solver approach for four well-known problems: The Traveling Salesperson Problem (TSP), the Uncapacitated Facility Location Problem (UFLP), the Maximal Set Cover Problem (MCP), and the Weighted Tardiness Problem (WTP). For each of the four problem types, we randomly generated 100 problem instances and ran both solution procedures using version 5 of the PSE. We did some preliminary experiments to find a problem size for which, on average, the problems did not solve in less than a second and did not take more than two minutes. (Details of the problem sizes and the sampling parameters are given in the appendix, together with descriptions of the models we used.) On all problems, we imposed a 5-minute maximum run time. This limit rarely caused any difficulty, except for one case which we discuss later. For integer programming, the Tolerance parameter was set to zero, thus ensuring that an optimal solution would be found unless the run time limit was reached.

We were also interested in the performance of version 6 of the Premium Solver Platform (PSP). The PSP handles much larger problem sizes and contains some algorithmic enhancements (Frontline Systems 2003). We noticed, however, that on the same problem sizes that provided interesting differences between the two approaches when we used the PSE, the PSP solved 398 out of 400 problems optimally using the evolutionary solver. We suspected that this near-perfect performance was mainly due to the problem sizes involved. Therefore, we carried out a second set of comparisons for the PSP using larger problem sizes.

Our experiments were implemented on laptop computers. The PSE results came from a machine with a 1.7 GHz processor and 512 MB of RAM. The PSP results came from a different machine with a 1.7 GHz processor and 1 GB of RAM.

The evolutionary solver requires the specification of several execution parameters, and we selected parameters that might be representative of student use. We set the Population Size at 200, which is the maximum permitted, and we took the Mutation Rate at its default value of 7.5%. These two choices appear to create a reasonable amount of diversity in the resulting population. We also kept the Convergence parameter at its default value of 0.0001. Used with the maximum possible population size, this setting controls the built-in termination condition (when 99% of the population achieves objective function values within 0.01%, the procedure terminates). Furthermore, we removed some other barriers by selecting large values for Max Subproblems and Max Feasible Solutions, so that these two ceilings would never cause termination within the 5-minute time limit. Then we set the Max Time without Improvement to 60 seconds with a Tolerance of zero. This choice meant that when a minute elapsed without any progress toward a better solution, the search was terminated. Thus, the time limit, the convergence limit, and the tolerance limit served as three parallel conditions for termination.

Normally, the evolutionary solver would be rerun with changes in the execution parameters. However, in our experiments, we treated the evolutionary solver as if it were a substitute for integer programming and observed the results of only one run. The advantage of this design was that we could conduct our experiments on large samples rather conveniently.

Following traditional practice in the research literature, we collected summary statistics on the following metrics:

(1) How often did the evolutionary solver produce an optimum?

(2) What was the worst suboptimal result?

(3) What was the average suboptimality?

(4) What was the average suboptimality among suboptimal outcomes?

Specifically, Question (1) was answered by the percentage of times that the evolutionary solver produced an optimum. For Questions (2)–(4) we measured suboptimality as the difference between the objective function values of the two approaches, taken as a ratio to the optimal solution. In the following subsections we provide results for each of the four problems.

The Traveling Salesperson Problem

A summary for the TSP appears in Table 1. The integer programming solutions were all obtained within the 5-minute time limit. All of the evolutionary solver runs were terminated by the 1-minute limit for the

Table 1 Results of the Computational Study for TSP

TSP	PSE (%)	PSP (%)
% ES optimal	83.0	16.0
Worst suboptimality	17.0	83.0
Average suboptimality	0.7	4.6
Cond. average suboptimality	3.9	5.5

Max Time without Improvement, with the exception of one run for the larger problem size. Only 83% of the evolutionary solver runs produced an optimum in the small problem set; for the large problem set, the figure was 16%. The deteriorating performance on the larger problems suggests that the evolutionary solver has considerable difficulty with the higher number of combinations in larger problems.

The Uncapacitated Facility Location Problem

A summary for the UFLP appears in Table 2. The integer programming solutions were all obtained within the 5-minute time limit. All of the evolutionary solver runs were terminated by the 1-minute limit for the Max Time without Improvement. However, the overall picture is rather different than for the TSP. In the case of the UFLP, the PSE generated optimal solutions in 92% of the problems, while in the larger problems, the PSP generated optimal solutions every time. Evidently, the evolutionary solver is very effective on the UFLP, even for large problem sizes, when the PSP is used.

The Maximal Set Cover Problem

A summary for the MCP appears in Table 3. Again, the integer programming solutions were all obtained within the 5-minute time limit. All of the evolutionary solver runs were terminated by the 1-minute limit for the Max Time without Improvement or by the Convergence condition. In these respects, the outcomes are reminiscent of the outcomes for the TSP. However, for the MCP, the optimality performance of the evolutionary solver is noticeably worse. Only 54% of the small problems were solved to optimality by the PSE, and only 5% of the large problems in the case of the PSP. Evidently, the MCP is quite a difficult problem for the evolutionary solver.

Table 2 Results of the Computational Study for UFLP

UFLP	PSE (%)	PSP (%)
% ES optimal	92.0	100.0
Worst suboptimality	4.1	0.0
Average suboptimality	0.1	0.0
Cond. average suboptimality	1.7	0.0

Table 3 Results of the Computational Study for MCP

MCP	PSE (%)	PSP (%)
% ES optimal	54.0	5.0
Worst suboptimality	69.3	46.0
Average suboptimality	11.4	4.2
Cond. average suboptimality	24.3	4.4

The Weighted Tardiness Problem

We were able to create WTP instances that were intrinsically "difficult" cases by focused sampling of processing times, due dates, and weighting factors. The results for the WTP problems appear in Table 4.

The evolutionary solver approach does a very good job of finding the optimal solution in these problems and in most cases terminates much faster than the integer programming approach. The evolutionary solver found optima in 98% of the small problems and 100% of the large problems. (The "large" problems were only incrementally larger than the small problems, but recall that we chose problem sizes according to average solution times observed for the slower of the two methods.) We used a convenient integer programming formulation that is relatively straightforward to build in a spreadsheet, but it is apparently not a tight formulation. In the large problems, there were 20 cases in which the integer program did not finish within 5 minutes. In 17 of these cases, the objective functions of the two approaches matched and in 3 cases, the ES solution was actually better than the integer programming solution on hand at the time it was terminated.

3. Student Choices and Performance

The picture that emerges from the computational results is somewhat complicated. The effectiveness of the evolutionary solver seems to depend on problem type as well as problem size. In some situations it is very effective, but in others it is not. What then should we teach students about the use of Solver? To begin answering this question, we should also know something about our students' preferences. Ideally, a student should learn to build models for the integer solver as well as models for the evolutionary solver.

Table 4 Results of the Computational Study for WTP

WTP	PSE (%)	PSP* (%)
% ES optimal	98.0	100.0
Worst suboptimality	1.1	0.0
Average suboptimality	0.0	0.0
Cond. average suboptimality	0.9	0.0

*Only 80 of the 100 problems were solved to optimality by the integer programming model in the allotted 5 minute run time. ES found the optimal solution in all of these cases.

Faced with an optimization problem, which modeling approach will the student choose? And how good a model will the student build? We had an opportunity to observe the results of a natural experiment that may shed some light on the answers to these questions.

The natural experiment was a take-home final exam given to 61 students in an Optimization course. The students were primarily Master's students in Engineering, many of whom were pursuing a degree in Engineering Management.[2] During the course, which emphasized the application of Premium Solver, the students were exposed to both approaches. In some cases, the students practiced both methods for solving the same problem, in order to compare the two approaches and explore the advantages and disadvantages of each. The reading and lectures did not take a general position on the desirability of using one approach or the other.

On the final exam, the students were presented with seven problems from which they were required to select four to solve. They were not told which approach to take or that either approach was possible, but a follow-up question asked them to indicate whether their result was a global optimum, a local optimum, or neither. This question served as a cue that optimality might be at issue.

Although the students were not told that each problem was amenable to solution by integer programming, it was possible to find a global optimum to each of the problems using the integer solver, employing principles that had been covered in the course. Nevertheless, 48% of the solutions were obtained with the evolutionary solver. Thus, the evolutionary solver had considerable appeal to this audience.

An optimal solution was obtained by 84% of the students who took an integer programming approach, while the figure for the students who took an evolutionary solver approach was only 71%. The errors in the integer programming approach were mostly due to formulation mistakes, although in a few cases there were typos and other minor model flaws. Some of the incorrect answers in the evolutionary solver approach occurred due to model flaws, but some occurred in valid models where the evolutionary solver could not produce an optimum under the various parametric settings that the students tested. (The exam was structured as a one-week take-home and therefore time pressure was not severe.)

The detailed results of the natural experiment are shown in Table 5. Three of the exam problems matched the problem types in our computational experiments. In the case of the TSP, the students had been assigned homework examples using both approaches. In the exam, the students preferred an integer programming approach (86%), and they were more likely to obtain the optimal tour length with integer programming. This outcome is consistent with the experimental results, which indicate that the TSP can be challenging for the evolutionary solver. Being familiar with both approaches, the students apparently preferred the integer programming approach because they were confident that they could find an optimum.

Table 5 Results of the Final Exam Natural Experiment

		Number		Correct answer	
9×9	IP	30	86%	28	93%
TSP	ES	5	14%	3	60%
11×4	IP	15	65%	12	80%
Makespan	ES	8	35%	6	75%
8-job Wtd.	IP	1	3%	1	100%
Tardiness	ES	30	97%	29	97%
Cutting	IP	21	55%	17	81%
Stock	ES	17	45%	6	35%
Max	IP	24	63%	21	88%
Bundles	ES	14	37%	11	79%
3×25 Max	IP	12	33%	12	100%
Cover	ES	24	67%	15	63%
Maximin	IP	15	56%	8	53%
Preferences	ES	12	44%	8	67%
Total	IP	118	52%	99	84%
	ES	110	48%	78	71%

The MCP was new; the students had not seen it during the course. In the exam, the students preferred the evolutionary solver approach (67%), but they were more likely to obtain the optimal coverage with integer programming. This outcome is also consistent with the experimental results, which indicated that the MCP was very challenging for the evolutionary solver.

The WTP was familiar, but the students had been assigned homework on this type of problem only with the evolutionary solver (although both methods had been covered in the reading). In the exam, only one student out of the 31 who attempted the WTP chose an integer programming approach. That student obtained a correct answer, but so did almost all of the students who relied on the evolutionary solver. Once more, this outcome is consistent the experimental results, which indicate that the evolutionary solver is very effective on the WTP.

When we examined the details of the remaining exam problems, we saw further patterns. The cutting stock problem had been used as an example of

[2] These students often resemble management majors more closely than engineering majors, for the purposes of drawing inferences about management students.

the evolutionary solver approach in class but had not ever been mentioned in conjunction with integer programming. Yet, the majority of the students who attempted this problem (which was more extensive than the class example) took an integer programming approach. In addition, those students were far more likely to find the optimum than the students who relied on the evolutionary solver. At first glance, this result suggests that integer programming will be the preferred choice if there is considerable doubt about the ability of the evolutionary solver to produce an optimum. However, the same conditions applied to the MCP, and there the students preferred the evolutionary solver. The cutting stock problem is a special case in which the spreadsheet model for integer programming is quite similar to the evolutionary solver model. Starting with a valid evolutionary solver model (which was nearly a given in this case), a student could convert to an integer programming model with modest effort and some insight. The same cannot be said of the MCP.

The makespan problem was analogous to an example from lecture that explained how to linearize the maximum of several functions. In the exam, students preferred an integer programming approach (65%) and were more likely to find the correct answer using that approach than were those who relied on the evolutionary solver. Exposure to an example integer program and some confidence in applying the integer solver were apparently more effective than the convenience offered by the evolutionary solver.

The bundling problem had virtually no roots in examples that appeared in lecture or in the reading. Here, the picture was typical: students preferred an integer programming approach (63%) and were more likely, as a result, to obtain the correct answer.

The outcomes were somewhat reversed in the maximin preference problem, which required students to find an optimal assignment of students to class projects. A key constraint required a minimum team size for any project, and the objective was to maximize the minimum satisfaction score among the assignments made. Thus, there were two formulation challenges for a valid integer programming model, neither of which is straightforward. In the exam, students still preferred the integer programming approach (56%), but they were less likely to locate an optimal assignment than those who used the evolutionary solver. Here, the formulation challenges in the integer programming approach presented more of an obstacle than the possible shortcomings of an evolutionary solver model. Nevertheless, only 67% of the evolutionary solver models reached an optimum.

These students often resemble management majors more closely than engineering majors, for the purposes of drawing inferences about management students.

4. Summary and Conclusions

We have provided computational results for integer programming and for the evolutionary solver using both the PSE (smaller problems) and the PSP (larger problems). Based on the problems we chose to study, and for the particular spreadsheet models we implemented, we have given an indication of the degree of suboptimality the user might experience with the evolutionary solver. Cleary, as teachers, we should communicate the possibility of suboptimality with evolutionary solver, even for correctly defined models, and this paper suggests the kinds of examples we might use as illustrations.

Our computational results also show that the evolutionary solver can be quite effective on certain problems, such as the WTP and the UFLP. Therefore, our teaching should also convey the message that the evolutionary solver can perform quite well on some problem types, but not on all. We need to learn a good deal more about the problem types on which the evolutionary solver performs well.

The exam experience suggests that students tend to choose an approach based on familiarity and confidence. When they are trained on both approaches and feel confident that they can implement an integer programming model correctly, that will most often be their preference because a correct implementation leads to an optimal solution. When students confront a problem where they lack practice or confidence in an integer programming approach, they may prefer the evolutionary solver. As teachers, we may need to be more careful about the kinds of practice that we give our students.

Although integer programming models are often difficult for students to build because formulations require non-intuitive ways of expressing problem features, we have observed that the evolutionary solver approach can be difficult as well. By accommodating sophisticated and sometimes unusual Excel functions, the evolutionary solver can create new challenges in model formulation. As teachers, we may need to steer students away from formulation approaches that are prone to error.

Some students seem vulnerable to a psychological trap when they use the evolutionary solver. A series of runs under different parameter settings that produces the same solution may induce students to conclude prematurely that they have found an optimum. As teachers, we need to develop better guidelines for implementing the evolutionary solver so that the chances of reaching an optimum improve.

In essence, the evolutionary solver approach is not necessarily simpler because it is more flexible. The evolutionary solver is a sophisticated tool, but it cannot substitute for an optimization model. To use the evolutionary solver in teaching, we may need a more

sophisticated pedagogical approach than our current level of knowledge supports.

Appendix. Alternative Models for the Four Problem Types

Traveling Salesperson Problem

Minimze the length of a tour that visits each of n cities once and returns to the starting city.

Notation

d_{ij} = distance from city i to city j (given)
x_{ij} = 1 if tour includes a link from city i to city j; 0 otherwise (binary decision variable)
u_i = relative sequence position for city i (integer decision variable)
$p(j)$ = immediate predecessor of city j in sequence with $p(1) = n$

Problem sizes

PSE problems: 10 cities PSP problems: 15 cities

Sampling

d_{ij} = drawn from a uniform distribution on (500, 5,000)

IP Formulation

$$\text{Minimize} \quad z = \sum_{ij} d_{ij} x_{ij}$$

$$\text{subject to:} \quad \sum_i x_{ij} = 1 \quad \text{for all cities } j$$

$$\sum_i x_{ij} = 1 \quad \text{for all cities } i$$

$$u_i - u_j + n x_{ij} \le n - 1, \quad \text{for all pairs } (i, j) \text{ not including city 1}$$

ES Formulation

$$\text{Minimize} \quad z = \sum d_{p(j)j}$$

$$\text{subject to:} \quad j = \textit{alldifferent}$$

Uncapacitated Facility Location Problem

Minimize the sum of fixed costs and variable costs for m potential facility locations and n customer demand locations.

Notation

F_i = fixed cost for facility i (given)
c_{ij} = variable distribution cost from facility i to location j (given)
d_j = demand at location j (given)
M = large positive number ($\sum dj$)
x_{ij} = quantity sent from facility i to customer location j (decision variable = 0)
y_i = 1 if facility i is used in the design; 0 otherwise (binary decision variable)

Problem sizes

PSE problems: 10×19 PSP problems: 15×75

Sampling

F_i = drawn from a uniform distribution on (1,000, 5,000)
c_{ij} = drawn from a uniform distribution on (0, 1)
d_j = drawn from a uniform distribution on (1,000, 9,000)

IP Formulation

$$\text{Minimize} \quad z = \sum_i F_i y_i + \sum_{ij} c_{ij} x_{ij}$$

$$\text{subject to:} \quad \sum_i x_{ij} \ge d_j \quad \text{for all locations } j$$

$$\sum_j x_{ij} \le M y_j \quad \text{for all facilities } i$$

ES Formulation

$$\text{Minimize} \quad z = \sum_i y_i \left[IF \sum_i y_i = 1, F_i, 0 \right] + \sum_j \text{MIN}[c_{ij} d_j \mid y_i = 1]$$

$$\text{subject to:} \quad y_i = 0 \text{ or } 1$$

Max Cover Problem

Maximize the demand volume serviced by a specified number (K) of facilities, each of which is capable of servicing a subset of the demand locations.

Notation

d_i = demand at location i
a_{ij} = 1 if facility i can service location j; 0 otherwise (given)
K = number of open facilities (given)
x_i = 1 if location i is serviced; 0 otherwise (decision variable, $0 \le x_{(i)} \le 1$)
y_j = 1 if facility j is used in the design; 0 otherwise (binary decision variable)

Problem sizes

PSE problems: 30×30 PSP problems: 100×100

Sampling

d_i = drawn from a uniform distribution on (1, 1,000)
a_{ij} = 1 with probability 0.15; 0 otherwise

IP Formulation

$$\text{Maximize} \quad z = \sum_i d_i x_i$$

$$\text{subject to:} \quad \sum_i a_{ij} y_j \ge x_i \quad \text{for all locations } i$$

$$\sum_i y_j \le k$$

ES Formulation

$$\text{Maximize} \quad \sum_i d_i \left[IF\left(\sum_j a_{ij} y_j > 0, 1, 0 \right) \right] - IF\left(\sum_j y_j > K, \infty, 0 \right)$$

$$\text{subject to:} \quad y_i = 0 \text{ or } 1$$

Weighted Tardiness Problem

Minimize the total weighted tardiness among n jobs sequenced at a single processor.

Notation

p_j = processing time for job j (given)
w_j = weighting factor for job j (given)
d_j = due date for job j (given)
s_j = start time for job j (decision variable = 0)
t_j = tardiness for job j (decision variable) = $\max\{s_j + p_j - d_j, 0\}$
y_{kj} = 1 if job j follows job k in sequence; 0 otherwise (binary decision variable)

M = large positive number ($\sum p_j$)
$x(j)$ = index of job in sequence position j (integer decision variable)

Problem sizes

PSE problems: 8 jobs PSP problems: 9 jobs

Sampling

p_j = drawn from a uniform distribution on (1, 25)
w_j = drawn from a uniform distribution on (1, 10)
d_j = drawn from a uniform distribution centered at $0.2\sum p_j$ with a range of $0.3\sum p_j$

(The sequencing literature calls this a tardiness factor of 0.8 and a due-date range of 0.3)

IP Formulation

$$\text{Minimize} \quad z = \sum w_j t_j$$

$$\text{subject to:} \quad s_j - s_k + My_{jk} \le M - p_j \quad \text{for all job pairs } j < k$$

$$s_j - s_k + My_{jk} \le M - p_k \quad \text{for all job pairs } j < k$$

$$s_j - t_j \le d_j - p_j \quad \text{for all jobs } j$$

ES Formulation

$$\text{Minimize} \quad z = \sum w_{x(j)} t_{x(j)}$$

$$\text{subject to:} \quad x(j) = \textit{alldifferent}$$

References

Camm, J. D. 1998. What do spreadsheets offer OR/MS majors? *INFORMS National Meeting*, Montreal.

Camm, J. D. 2000. OR in pictures. *INFORMS National Meeting*, San Antonio, Texas.

Erkut, E. 1998. How to "Excel" in teaching management science. *OR/MS Today* **25**(5) 40–43.

Frontline Systems. 2003. *Premium Solver Platform V5.5 User Guide*.

Ragsdale, C. T. 2001. Teaching management science in spreadsheets, from decision models to decision support. *INFORMS Trans. Ed.* **1**(2) http://ite.pubs.informs.org/Vol1No2/Ragsdale/Ragsdale.php.

Savage, S. 2001. Blitzograms, interactive histograms. *INFORMS Trans. Ed.* **1**(2) http://ite.pubs.informs.org/Vol1No2/Savage/Savage.php.