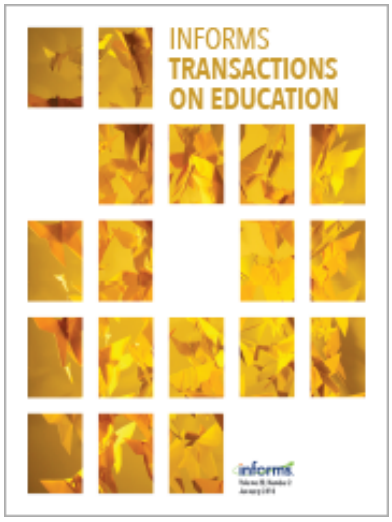


INFORMS Transactions on Education

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

Using Board Puzzles to Teach Operations Research

Gail W. DePuy, G. Don Taylor,

To cite this article:
Gail W. DePuy, G. Don Taylor, (2007) Using Board Puzzles to Teach Operations Research. INFORMS Transactions on Education 7(2):160-171. https://doi.org/10.1287/ited.7.2.160

Vol. 7, No. 2, January 2007, pp. 160–171
ISSN 1532-0545 | 07 | 0702 | 0160
INFORMS Transactions on Education
DOI 10.1287/ited.7.2.160

Using Board Puzzles to Teach Operations Research

Gail W. DePuy
Department of Industrial Engineering, University of Louisville, Louisville, Kentucky 40292,
depuy@louisville.edu

G. Don Taylor
Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University,
Blacksburg, Virginia 24061, don.taylor@vt.edu

Engineering design is a difficult task. Designers work at the boundaries between physical laws, societal or governmental regulations, the need for efficiency, and the desire to develop creative or aesthetic solutions. Teaching engineering design is therefore a difficult task. How is it possible to stimulate students to think of creative solutions while adhering to established principles and necessary rules? Certainly, it is important for students to have deep-seated roots in engineering fundamentals. These fundamentals include both a strong overall engineering core and coursework specific to the particular branch of engineering study undertaken by the student. For industrial engineers, the study of operations research is fundamental in preparing them for careers in engineering design. This paper addresses one way of encouraging creativity in teaching operations research in industrial engineering design curricula. Specifically, this paper provides examples of the use of games as pedagogical tools in teaching operations research. These games cannot be used as a substitute for teaching practical applications, but they are excellent supplementary tools that encourage students to tackle highly structured problems creatively while working on interesting and enjoyable tasks. Four examples are provided within the paper.

1. Introduction

Recent research supports the idea that students and employees are happier and more motivated when they enjoy their studies or their work. According to Meyer (2000) a growing number of US companies are attempting to make work more "light-hearted" through the use of games, trips, and other on-the-job perks. Meyer's work indicates that doing so helps in job recruiting and retention, boosts morale and opens up new ways to communicate in the workplace. Similarly, in more of a teaching and learning setting, Rogers and Williams (1993) make use of the experiences of quality gurus, teachers, and supervisors to suggest that "fun" enhances both learning and productivity.

One can make learning more fun by using games to support engineering courses or other curricula about the complicated process of engineering design. It should be noted that games should not replace, but simply supplement the use of more realistic engineering design applications in the classroom. In the same way that companies use perks to satisfy employees, engineering educators can use games to add interest and fun to the learning process.

It can be relatively simple to teach engineers the use of mathematical, scientific, or related concepts and techniques, but it is very difficult to teach engineers to use these tools and techniques creatively in design activities. One way to teach creativity is to assign problems that would not normally be considered viable candidates for the solution methodology. In this context, games may be particularly useful in teaching creativity in engineering design. This is often evident in teaching mathematical programming techniques to industrial engineers. It seems that students are well-prepared to solve problems once formulated, but most engineering educators would likely agree that the more creative task of problem formulation is the weak link in the educational chain. Formulating a solution to a game using mathematical programming can require a great deal of creative thought, and is interesting and fun at the same time. The formulations are easily visualized, and the solutions are straightforwardly verified using simple games of the types discussed herein. It is thought the formulation skills gained are easily transferable to more realistic engineering problems although this area of research needs further investigation.

This paper is organized as follows. A review of literature related to using games and puzzles for educational purposes is presented in the next section. Section 3 discusses the literature which addresses the use of puzzles and games to teach operations research (OR). Sections 4, 5, and 6 present the formulations for several puzzles to be used in an OR course as well as discuss the authors' experiences using these puzzles within their undergraduate and graduate courses. Finally concluding remarks are presented in §7.

2. Supporting Literature

The benefits of using toys and games for teaching young children have been known for quite some time. See Farkas (1993) or Bottomley and Parry (1998) for examples. In the study by Farkas (1993), the focus is on using toys and games in conjunction with computer equipment to help develop children's thinking ability. Bottomley and Parry (1998) use playground equipment to teach and to differentiate between physics and engineering. In a more general sense, Heineke and Meile (1996) and Nafalski et al. (1994) discuss how games can be used to introduce new topics, reinforce conceptual learning, and provide common experiences to positively impact both teaching and complex learning. The benefit of active learning in higher education through the use of games is addressed by Cowan (1998). Other authors discuss the games they use more specifically. Ermer and Phipps (1996), for example, use Tinkertoys and egg drop contests to teach planning, communication, and teamwork. Zafran (1995) makes use of laser-based games to concurrently teach optics and teamwork. Also of interest is a paper by Carswell and Benyon (1996) in which the authors describe how an adventure game can be used to support distance education and to better motivate off-campus students that can feel isolated from the learning process.

A survey of 174 articles related to teaching engineering design is presented by Dym et al. (2005) with several included authors employing games. The majority of literature related to teaching engineering design comes from the field of computer science. Jones (2000) works with students in designing and implementing computer games as a primary deliverable in his undergraduate capstone course in computer science at Colby College. Similarly, Pavlidis (1997) uses video games to teach computer graphics. Hill et al. (2003) advocate the use of puzzles and games as a way of addressing different learning styles when teaching computer operating systems concepts. Levitin and Papalaskari (2002) discuss the use of puzzles in teaching design and analysis of algorithms. The use of simulation games to assist in teaching statistics and experimental design has also been documented (Asao 1990, Schwarz 2003). Sniedovich (2002a) address several benefits of using games to teach operations research and serves as motivation for the work presented in this paper.

3. The Use of Games in Teaching Design Through Operations Research

While operations research contains key design tools for industrial engineering students, students often have a difficult time learning the basic concepts. As mentioned previously, perhaps the greatest difficulty for students is to learn problem formulation techniques. Part of the formulation difficulty stems from the fact that many of the industrially relevant problems are ill-defined and difficult to conceptualize. This is particularly true when students are asked to formulate problems to assist in the design and development of complex interactive systems. Games, and especially familiar games, are very useful in assisting students in visualizing problems, breaking problems down into manageable sub-sections, developing solution strategies, and seeking known and well-established goals. These skills, once learned, are transferable to other applications.

Another issue regarding industrially relevant problems is that the basic configurations of many of these problems have already been formulated and solved by other researchers and practitioners, and they appear in many textbooks. While these readily available formulations are important to teach and know, they do not assist in teaching the "creative process" of design. Furthermore, the remaining related problems are often those that are difficult to formulate or solve because they are simply too large to solve using basic operations research techniques. Games provide a platform for teaching creativity, because few teachers have employed them as teaching tools and even fewer have made use of mathematical programming to solve them. It is easy to find examples of fun and interesting games that are can be tractably solved via mathematical programming and that teach skills that are immediately transferable to realistic problem settings. The practice of using puzzles/games to reinforce the teaching of operations research has appeared in the literature. The puzzles and games addressed include the riddle of the pilgrims (Chlond 2002a), towers of Hanoi (Sniedovich 2002b), the traveling space telescope problem (Chlond 2002c), fiveleapers a-leaping (Chlond et al. 2003), egg drop (Sniedovich 2003b), flip (Trick 2001), nimatron (Chlond and Akyol 2003), logical puzzles (Chlond and Toase 2003), and the counterfeit coin problem (Sniedovich 2003a). Chlond and Toase (2002), Letavec and Ruggiero (2002), and Foulds

and Johnson (1984) present mathematical formulations for several chessboard placement puzzles. We have found that these chessboard puzzles and their resulting formulations serve as an excellent introduction to linear/integer programming, because they are relatively straightforward.

This paper presents several puzzle games appropriate for use in both undergraduate and graduate operations research courses. The next three sections introduce different puzzle games, present their formulations, and discuss the authors' experience using these games in the classroom.

4. The Cracker Barrel Peg Board Problem

The Cracker Barrel peg game is a very interesting game for teaching engineering design using operations research. This game is commonly found on the tables of Cracker Barrel Old Country Stores. The game involves "jumping" pegs over other pegs into empty holes. The jumped peg is then removed from the board. The objective is to be recognized as a "genius" by leaving only one remaining peg at the end of the game. A picture of the game appears in Figure 1. Cracker Barrel (2006a) provides a brief history of the peg game. The game can be played interactively on the Internet (Cracker Barrel 2006b).

4.1. Cracker Barrel Formulation

The formulation for the Cracker Barrel game involves two sets of binary decision variables; one represents peg moves and the other indicates board status after each move. Due to the layout of the board, identifying the origin and destination position of a peg jump determines the position jumped. For example, for the peg hole labeling diagram shown in Figure 2, knowing a peg was moved from position M to position H indicates position K was jumped. The formulation shown below only generates variables for feasible moves as defined by the problem parameters.

Figure 1 Cracker Barrel Peg Board Game

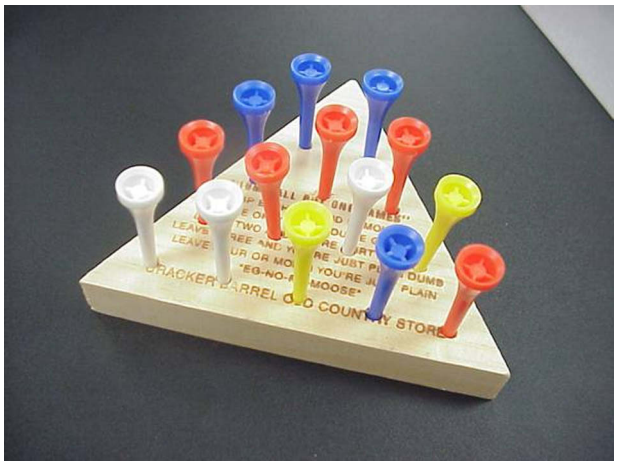
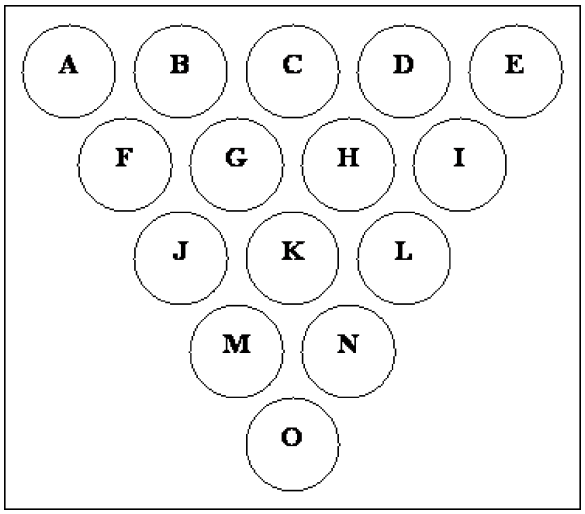


Figure 2 Hole Labeling Diagram for Peg Board Game



Decision Variables

$$X_{ik} = \begin{cases} 1 & \textit{if position } i \textit{ empty at end of move } k \\ 0 & \textit{if peg in position } i \textit{ at end of move } k \end{cases}$$

$$Y_{ijk} = \begin{cases} 1 & \textit{if peg moved from position } i \textit{ to position } j \\ & \textit{during move } k \\ 0 & \textit{otherwise.} \end{cases}$$

The board starts with pegs in 14 of 15 holes. Therefore, 13 moves will be required to go from the 14 initial pegs to the one final peg, which is the goal of this puzzle. Consequently there will be $15 * 13 = 195$ X_{ik} variables generated. Similarly, there are 36 initially valid moves for this puzzle, therefore $36 * 13 = 468$ Y_{ijk} variables are generated in this formulation.

Problem Parameters

$D_i =$ set of feasible "destination" or "jump to" positions for a peg in position i
$O_i =$ set of feasible "origin" or "jump from" positions for a peg in position i
$E_i =$ set of (origin, destination) position pairs that jump over position i

Problem Formulation

Objective Function:

Maximize number of empty holes at end of 13 moves

$$\text{Maximize} \sum_i X_{1,13} \qquad (1)$$

Subject to:

One jump per move $$\sum_i \sum_{j \in D_i} Y_{ijk} = 1 \quad \forall k \tag{2}$$

Initial conditions $$X_{i,0} \in \{0,1\} \quad \forall i \tag{3}$$

Can only jump to empty hole

$$\sum_{j \in O_i} Y_{jik} \le X_{ik} \quad \forall i,k \tag{4}$$

Can only jump from hole with peg

$$\sum_{j \in D_i} Y_{ijk} \le 1 - X_{ik} \quad \forall i,k \tag{5}$$

Can only jump over peg

$$\sum_{(j,i) \in E_i} Y_{ijk} \le 1 - X_{ik} \quad \forall i,k \tag{6}$$

Balance equations

$$X_{ik} + \sum_{(j,i) \in E_i} Y_{jik} + \sum_{j \in D_i} Y_{ijk} - \sum_{j \in O_i} Y_{jik} = X_{i,k+1} \quad \forall i,k \tag{7}$$

Binary integer variables

$$Y_{ijk} \in \{0,1\} \quad \forall i, j \in D_i, k \tag{8}$$

$$X_{ik} \in \{0,1\} \quad \forall i,k \tag{9}$$

The objective function in Equation (1) maximizes the number of empty holes at the end of 13 moves. The constraints in (3) define starting conditions. Note that only one X_{i0} value can be initially equal to 1, indicating a starting condition with only one empty hole. The balance Equations (7) are the most conceptually difficult part of the formulation. These equations ensure that during each move period and for each hole, the hole status is equal to the initial hole status plus any changes made during the move. One unit is added to the hole status (i.e., it is made empty) if it is jumped or if a peg departs from it during the period. Similarly, one unit is subtracted (i.e., it is made full) if it is jumped into during the period.

4.2. Cracker Barrel Peg Board Solution

The solution to the problem depends upon the selection of the initial empty position as specified in the constraints in Equations (3) above. Generally speaking, the game has four initial starting scenarios with all other starting scenarios being a mirror image of others. The four possible starting scenarios are defined by the position of the initial empty peg hole.

- Corner (empty starting position at hole A, E, or O).
- Side (empty starting position at hole C, J, or L).
- Middle-Side (empty starting position at hole B, D, F, I, M, or N).
- Interior (empty starting position at hole G, H, or K).

Table 1 contains an overview of solutions provided by the IP model presented in Equations (1) through (9) for each of the possible starting conditions. Note that it is possible, for each possible starting hole, to leave only one remaining peg. Also, multiple optimal solutions exist to the problem for some of the starting scenarios. Although the IP finds optimal solutions, it does not find all possible optimal solutions for a given set of starting conditions. The run time for one starting scenario of this example using CPLEX 9.0.2 on an AMD OPTERON computer was 9.4 seconds.

4.3. Use of Cracker Barrel Peg Board Puzzle in OR Class

The Cracker Barrel Peg Board Puzzle was assigned as a two-week project to students in an undergraduate operations research course taught through an industrial engineering department. The project assignment given to the students briefly described the Cracker Barrel problem, directed the students to the interactive Internet site, and included the following hint: "there may be more than one way to formulate this puzzle, but a hint that may help you get started is to formulate the puzzle using two sets of decision

Table 1 Overview of Solutions to Peg Board Game

Starting empty hole	Corner (O)	Middle-side (M)	Side (J)	Interior (K)
1st Move	Jump L to O over N	Jump F to M over J	Jump C to J over G	Jump B to K over G
2nd Move	Jump J to L over K	Jump C to J over G	Jump I to G over H	Jump I to G over H
3rd Move	Jump D to K over H	Jump A to C over B	Jump E to C over D	Jump N to I over L
4th Move	Jump I to N over L	Jump D to B over C	Jump N to I over L	Jump J to L over K
5th Move	Jump B to D over C	Jump M to F over J	Jump B to D over C	Jump I to N over L
6th Move	Jump A to J over F	Jump L to C over H	Jump J to L over K	Jump A to J over F
7th Move	Jump O to L over N	Jump C to A over B	Jump L to E over I	Jump O to L over N
8th Move	Jump M to F over J	Jump A to J over F	Jump E to C over D	Jump D to B over C
9th Move	Jump E to C over D	Jump E to L over I	Jump O to J over M	Jump M to F over J
10th Move	Jump C to J over G	Jump N to I over L	Jump F to M over J	Jump F to H over G
11th Move	Jump F to M over J	Jump J to L over K	Jump C to J over G	Jump L to C over H
12th Move	Jump L to J over K	Jump I to N over L	Jump M to F over J	Jump B to D over C
13th Move	Jump J to O over M	Jump O to L over N	Jump A to J over F	Jump E to C over D
Final peg position	O	L	J	C

variables: one representing peg moves and the other indicating the board status after each move." The students taking this course, primarily juniors, worked in groups of two and submitted a written report showing their formulations and solutions for all possible starting configurations.

After, students were asked to anonymously complete a brief written survey containing the following questions:

1. What did you like about the Cracker Barrel project?
2. What did you dislike about the Cracker Barrel project?
3. Did the Cracker Barrel project help you visualize the solution?
4. Did the Cracker Barrel project help you to develop problem formulation skills?
5. Overall, was your understanding of math programming enhanced by use of this Cracker Barrel project?

The vast majority of responses were favorable. Of the 18 students taking the course, 17 responded positively to questions 3, 4, and 5. Selected comments include: "I learned so much from it. It's much more helpful than homework"; "Yes (the project helped develop problem formulation skills). It took a lot of thought to represent the problem mathematically"; "It was very challenging and actually became addictive"; "I liked that I could try my solution to see if I had the right formulation"; "Interesting project since it's a game everyone has played." The responses to question 2 include: "I spent lots of time on this project"; "Need additional labs to gain more practice using (the solution software)"; "It was difficult. It was unlike other types of problems we had concentrated on."

This assignment was incorporated into the operations research course as a means of introducing the students to two main concepts. First, because this formulation will require many more decision variables than the typical homework formulation, this assignment serves as a good introduction to large-scale optimization problems and can fuel discussions of both the size of and the solution techniques for formulations of real-world problems. Second, because this puzzle requires students to link the values of decision variables from one time period (i.e., move) to the next, it serves as a good introduction to a variety of other classic problems with similar timing traits in areas such as inventory control and scheduling.

5. Peg Solitaire Problem

There are a variety of peg solitaire games in which pegs are arranged in an initial configuration on a holed board, a series of peg moves are made where the jumped peg is removed, and the objective is to arrive at the desired final board configuration. The variations of peg solitaire come from the shape of the board (number and arrangement of holes), the initial peg configuration, and the final peg configuration. Figure 3 shows several peg solitaire board shapes. The standard peg solitaire board consists of 33 holes arranged as shown in Figure 4. The standard initial configuration, also known as Hi-Q, consists of pegs in all positions except the center position (as shown in Figure 3). However, other initial configurations exist, with descriptive names such as cross, plus, fireplace, arrow, double arrow, diamond, and pyramid. The standard peg solitaire game dates to the late 17th century (Elliot Avedon Museum and Archive of Games 2006) and there has been much discussion of these games in the literature including a variety of solution techniques (Beasley 1992, Berlekamp et al. 1982, Gardner 1969, Moore and Eppstein 2002, Uehara and Iwata 1990, Kraitchik 1942, Guy 1998) although an integer programming approach such as the one presented here was not found.

The Cracker Barrel game presented in the preceding section is a special form of the peg solitaire game and is alternatively called triangular peg solitaire. Therefore the formulation developed in the preceding section can be extended to the general peg solitaire game.

5.1. Peg Solitaire Formulation

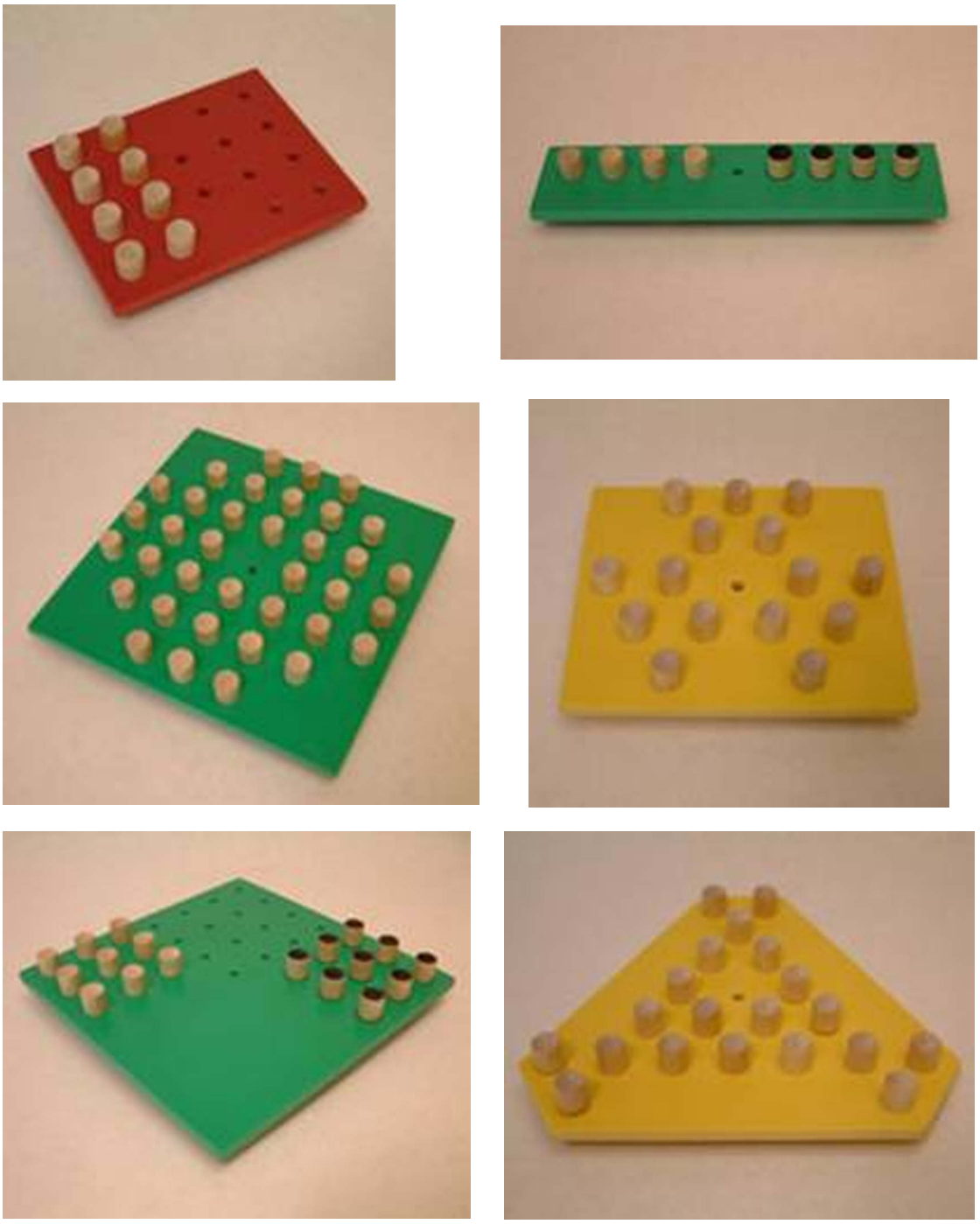
With minor modifications, the Cracker Barrel formulation presented in §4.1 can be used to solve any peg solitaire problem. Because of the different board shapes and initial board configurations of the peg solitaire problem, a different number of peg moves will be required to reach the solution. Because the jumped peg is removed on each move, the number of moves, N, is the difference between the number of pegs in the initial configuration and the final configuration (usually a single peg in the center hole). Therefore the objective function must be modified to reflect the number of required moves, and it is not necessarily 13 as was the case with the Cracker Barrel problem (see Equation (1a) below). Also, constraints (Equation (10)) must be added to define the desired final board configuration, where N is the number of moves to solve the puzzle. As mentioned previously, the typical final configuration is a single peg in the center position, but other final configurations could be desired. Also, the values of the problem parameters, D_i, O_i, and E_i, presented in §4.1 are clearly specific to the board shape and number of holes.

Maximize number of empty holes at end of N moves

$$\text{Maximize} \sum_i X_{i,N} \tag{1a}$$

$$\text{Final conditions} \quad X_{i,N} \in \{0,1\} \quad \forall i \tag{10}$$

Figure 3 Various Peg Solitaire Board Shapes

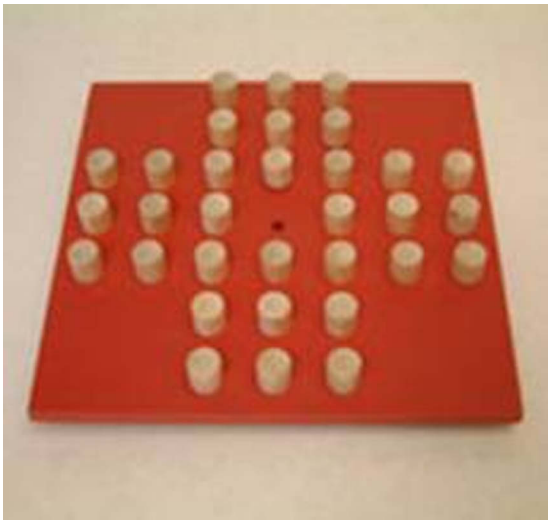


Therefore, the final formulation for the peg solitaire problem will include objective function (1a) and constraints (2)–(10).

5.2. Peg Solitaire Solution

As mentioned previously, many board shapes and initial configurations exist for peg solitaire, and all configurations can be solved using the formulation provided above. Mazeworks (2006) provides a Web peg solitaire game that includes several initial configurations for the standard 33-hole board shown in Figure 4. As an example, such a board with a cross initial configuration will be solved here. The peg hole labeling diagram and initial configuration are shown in Figure 5 (i.e., pegs in the shaded positions). The desired final configuration is a single peg in position 17. As the initial configuration has 6 pegs and the final configuration has 1 peg, the game will require

Figure 4 **Standard Peg Solitaire Game**



$N = 5$ moves. For this particular starting configuration there will be $33 * 5 = 165$ X_{ik} variables generated. In general, there are 76 valid moves for this standard 33-hole board. Therefore $76 * 5 = 380$ Y_{ijk} variables are generated in this formulation.

Table 2 shows the solution to the Figure 5 puzzle as found using the peg solitaire formulation presented in §5.1. The run time for this example using CPLEX 9.0.2 on an AMD OPTERON computer was 0.01 seconds.

There are many interesting modifications to the standard peg solitaire game (Beasley 1992). Two such modifications will be presented here: reverse peg solitaire and unconstrained peg solitaire.

5.3. Reverse Peg Solitaire

Reverse peg solitaire is similar to peg solitaire except the moves are in reverse; a jump over an empty hole fills it with a peg. The initial configuration is a peg in the center hole, a series of peg moves are made where a peg is added to the jumped empty hole, and the objective is to reach some desired final configuration such as a peg in all positions except the center, a cross, a plus, a fireplace, an arrow, a double arrow, a diamond, or a pyramid. The reverse peg solitaire game can be played at the Mazeworks (2006) site.

The previous peg solitaire formulation can be modified for this reverse peg solitaire problem. Constraints (2)–(5) and (8)–(10) from previous formulations are used, while the objective function and constraints (6) and (7) must be modified as shown in (1b), (6a), and (7a) for the reverse peg solitaire problem. Constraints (7a) update the status of each position for each move by adding one unit to the position status (i.e., making the position empty) if a peg departs the position; while one unit is subtracted (i.e., the position is made full) if the position is jumped or is the destination position. Similar to regular peg solitaire, the different final board configurations will require a different number of peg moves to reach the solution. For reverse peg solitaire, the number of moves, N, is the difference between the number of pegs in the final configuration and the initial configuration (usually 1).

Figure 5 **Peg Solitaire Game—"Cross" Configuration**

		1	2	3		
		4	5	6		
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
		28	29	30		
		31	32	33		

Table 2 **Solution to Example Peg Solitaire Puzzle**

1st Move	Jump 10 to 12 over 11
2nd Move	Jump 24 to 10 over 17
3rd Move	Jump 9 to 11 over 10
4th Move	Jump 12 to 10 over 11
5th Move	Jump 5 to 17 over 10

Minimize number of empty holes at end of N moves

$$\text{Minimize} \sum_i X_{i,N} \tag{1b}$$

Can only jump over empty hole

$$\sum_{(j,i)\in E_i} Y_{jik} \le X_{ik} \quad \forall i,k \tag{6a}$$

Balance equations

$$X_{ik} - \sum_{(j,i)\in E_i} Y_{jik} + \sum_{j\in D_i} Y_{ijk} - \sum_{j\in O_i} Y_{jik} = X_{i,k+1} \quad \forall i,k \tag{7a}$$

As an example, the reverse peg solitaire formulation presented above is used to solve the problem of starting with a single peg in the center position and performing a series of moves to reach the plus configuration (see Figure 6). The solution to this example is shown in Table 3. The run time using CPLEX 9.0.2 on an AMD OPTERON computer was 0.09 seconds.

5.4. Unconstrained Peg Solitaire

Another interesting extension of the peg solitaire game is unconstrained peg solitaire (Beasley 1992). As discussed in Chlond (2002b), the unconstrained

Figure 6 **Peg Solitaire Game—"Plus" Configuration**

		1	2	3		
		4	5	6		
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
		28	29	30		
		31	32	33		

Table 3 Solution to Example Reverse Peg Solitaire Puzzle

1st Move	Jump 17 to 19 over 18
2nd Move	Jump 18 to 16 over 17
3rd Move	Jump 17 to 29 over 24
4th Move	Jump 16 to 14 over 15
5th Move	Jump 15 to 17 over 16
6th Move	Jump 17 to 5 over 10
7th Move	Jump 16 to 18 over 17
8th Move	Jump 14 to 16 over 15

peg solitaire problem allows any integer number of pegs, positive, negative, or zero in each hole. Therefore, moves do not have to originate in occupied holes or terminate in unoccupied holes. As with regular peg solitaire, a peg is moved from an origin position to a destination position, and a peg is removed from the position jumped. The objective of unconstrained solitaire is to move from the initial configuration to the final configuration through a series of peg jumps. As noted by Chlond, the order in which the moves are carried out does not matter; the number of jumps to, from and over a position will determine its final status. Chlond includes a spreadsheet approach to solve the unconstrained peg solitaire problem but does not present a mathematical formulation. A formulation based on the peg solitaire notation introduced earlier in this section is now presented along with an example problem and results.

Decision Variables

Z_{ij} = number of jumps from position i to position j

Problem Parameters

D_i = set of feasible "destination" or "jump to" positions for a peg in position i

O_i = set of feasible "origin" or "jump from" positions for a peg in position i

E_i = set of (origin, destination) position pairs, which jump over position i

S_i = number of pegs in position i in initial configuration

F_i = number of pegs in position i in final configuration

Problem Formulation

Objective Function: Minimize number of moves

$$\text{Minimize} \sum_i \sum_{j \in D_i} Z_{ij} \tag{11}$$

Subject to:

Balance equations

$$F_i = S_i - \sum_{j \in D_i} Z_{ij} + \sum_{j \in O_i} Z_{jk} - \sum_{(j,k) \in E_i} Z_{jk} \quad \forall i \tag{12}$$

Integer variables

$$Z_{ij} \in \{-\infty, \ldots, -1, 0, 1, \ldots + \infty\} \quad \forall i, j \in D_i \tag{13}$$

Table 4 Solution to Unconstrained Peg Solitaire Puzzle Example

From	To	# moves	From	To	# moves
1	3	1	18	6	2
3	11	2	21	23	1
4	16	1	21	7	1
5	17	1	24	26	3
6	18	1	25	27	1
7	21	1	26	24	1
7	9	1	27	25	3
9	23	1	28	16	2
12	26	1	31	23	1
13	27	1	32	24	1
16	28	2	33	25	1
16	14	1			

As an example of the unconstrained peg solitaire problem, the unconstrained standard peg solitaire problem will be used. The initial configuration of the standard unconstrained solitaire consists of one peg in each position except the center position, which contains zero pegs. The final configuration of the standard unconstrained solitaire consists of zero pegs in each position except the center position, which contains one peg. Table 4 shows the number of moves between positions needed to arrive at the final configuration as determined by the math program formulated above. As stated previously, the order in which the Table 4 moves are performed does not matter. The run time for this example using CPLEX 9.0.2 on an AMD OPTERON computer was 0.09 seconds.

5.5. Use of Peg Solitaire Problems in OR Class

As the peg solitaire, reverse peg solitaire, and unconstrained peg solitaire problems are similar in nature to the Cracker Barrel peg board puzzle discussed in §4, they can be used in a similar fashion in the classroom. The authors plan to introduce these peg solitaire problems to students as alternatives to the Cracker Barrel assignment for an undergraduate operations research course. These peg solitaire problems will be used to introduce the students to the formulation and solution of large-scale optimization problems and time-based decision variables.

6. The Rush Hour Problem

The game "Rush Hour" by Binary Arts is also an interesting problem to assist in teaching operations research formulation and solution techniques. Once again, the problem is fun and challenging. The reader is referred to the ThinkFun, Inc. (2006) Web site to learn more about the game and the company that produces it. The Rush Hour game can be played interactively on the Internet at Baxter's (2006) Web site. A picture of the game appears in Figure 7.

Figure 7 Rush Hour Game

The objective of the game is to drive the target car (identified in Figure 7 by an arrow) out of the slot on the right side of the game board. In initial conditions, the path of the target car is blocked by other cars and trucks. Vehicles cannot turn, hence, they move either horizontally or vertically. The starting scenario is depicted on cards, similar to the one shown at the top of Figure 7. The game includes a number of cards describing game starting conditions ranging in difficulty. To drive the target car out of the slot, the other vehicles must be moved out of the way. They, in turn, are blocked by other vehicles. A number of complicated interaction effects must be considered and defeated to find a solution.

6.1. Problem Formulation

The formulation for the Rush Hour game is similar to that of the games presented in previous sections. A variable is needed to represent vehicle moves, and another is needed for board status. The formulation presented here makes a distinction between two board status variables. One binary integer variable, Z, takes on a value of 1 if a board position is occupied by a vehicle. Cars occupy 2 positions, while trucks occupy 3 positions. Another binary integer variable, X, is used to designate the position of each vehicle's "marker" or its most southwest (i.e., lower left) position. This marker is used to facilitate moving the vehicles and updating the board position of a vehicle after a move. The board position numbering scheme used in this formulation is shown in Figure 8. Note the exit position is located at position 18.

Figure 8 Rush Hour Board Position Numbering Scheme

1	2	3	4	5	6	
7	8	9	10	11	12	
13	14	15	16	17	18	EXIT
19	20	21	22	23	24	
25	26	27	28	29	30	
31	32	33	34	35	36	

Decision Variables

$$X_{i,j,k} = \begin{cases} 1 & \textit{if vehicle } i \textit{ marker occupies position } j \textit{ after move } k \\ 0 & \textit{otherwise} \end{cases}$$

$$Z_{i,j,k} = \begin{cases} 1 & \textit{if vehicle } i \textit{ occupies position } j \textit{ after move } k \\ 0 & \textit{otherwise} \end{cases}$$

$$Y_{i,j,l,k} = \begin{cases} 1 & \textit{if vehicle } i \textit{ marker moves from position } j \textit{ to position } l \textit{ during move } k \\ 0 & \textit{otherwise} \end{cases}$$

Problem Parameters

V_i = length of vehicle i
$m_{i,j}$ = set of positions occupied by vehicle i when its marker is in position j
$p_{j,l}$ = set of all positions between position j and position l (horizontal or vertical)
N = maximum number of moves

Problem Formulation

Objective Function:

Minimize number of moves

$$\text{Min} \sum_i \sum_j \sum_l \sum_k Y_{i,j,l,k} \tag{14}$$

Subject to:

Define winning (end of game) when target car (car #1) at exit position 18 on last move

$$X_{1,18,N} = 1 \tag{15}$$

Maximum of one vehicle move per turn

$$\sum_i \sum_j \sum_l Y_{i,j,l,k} \le 1 \quad \forall k \tag{16}$$

Update marker information for moved vehicle

$$X_{i,j,k-1} - \sum_l Y_{i,j,l,k} + \sum_l Y_{i,l,j,k} = X_{i,j,k} \quad \forall i,j,k \tag{17}$$

Define all positions occupied by a vehicle

$$V_i X_{i,j,k} \le \sum_{m \in m_{i,j}} Z_{i,m,k} \quad \forall i,j,k \tag{18}$$

Only one vehicle in a position at a time

$$\sum_i Z_{i,j,k} \le 1 \quad \forall j,k \tag{19}$$

Vehicles cannot move through or over occupied positions (i.e., other vehicles)

$$Y_{i,j,l,k} \le 1 - \sum_{i' \ne i} Z_{i',p,k-1} \quad \forall i,j,l,k,p \in p_{j,l} \tag{20}$$

Initial conditions $X_{i,j,0} \in \{0,1\} \quad \forall i,j$ (21)

$$Z_{i,j,0} \in \{0,1\} \quad \forall i,j \tag{22}$$

Binary integer variables

$$X_{ijk}, Z_{ijk} \in \{0,1\} \quad \forall i,j,k \tag{23}$$

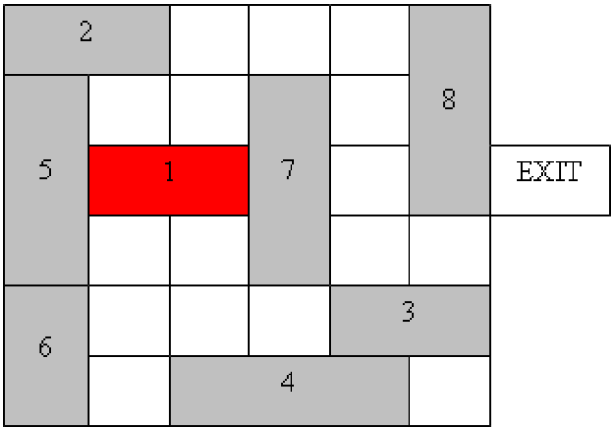
$$Y_{ijlk} \in \{0,1\} \quad \forall i,j,l,k \tag{24}$$

The objective function in Equation (14) minimizes the number of vehicle movements at the end of N moves. An arbitrary value of N must be selected by the user or student such that it is large enough to allow a sufficient number of moves but not too large, or it will dramatically increase the size of the IP to be solved and therefore, the runtime. Based on experimentation, using $N = 4*$(total number of vehicles) seems to work well. The balance Equations (17) update the marker information after each move. These equations ensure that during each move and for each car, the position status is equal to the initial position status plus any changes made during the move. One unit is added to the position status (i.e., it is made occupied) if a car's marker is moved to the position. Similarly, one unit is subtracted (i.e., it is made unoccupied) if a car's marker is moved from the position. Based on a vehicle's marker position, constraints (18) are used to indicate all board positions occupied by the vehicle. Constraints (20) ensure a vehicle move can only be made if the board positions between the origin and destination position are unoccupied (i.e., $Z_{ijk} = 0$). Constraints (21) and (22) are used to designate the board positions occupied in the initial scenario.

6.2. Rush Hour Solution

The initial scenario for an example Rush Hour puzzle is shown in Figure 9. The vehicles are arbitrarily numbered with the exception of the target car, which is number 1. Table 5 shows the solution for the puzzle as found using the formulation presented in §6.1. Obviously, the size of the formulation depends on the number and size of vehicles, which then, in turn, determine the number of possible moves. For example, in Figure 9, there are two possible positions for the vehicle 5 marker: positions 19 and 13. The formulation for this example, with $N = 12$, resulted in

Figure 9 Example Initial Scenario for Rush Hour Puzzle



1,970 decision variables and 5,043 constraints. The run time for this example using CPLEX 9.0.2 on an AMD OPTERON computer was 6.6 seconds.

6.3. Use of Rush Hour Problem in OR Class

This Rush Hour game was assigned as a three-week project to students in a graduate operations research course taught through an industrial engineering department. The project assignment contained a brief description of the Rush Hour problem, gave three initial scenarios to be solved, and directed the students to the Internet site. After allowing the students one week to develop their own formulations, the following statement was offered as a hint: "there may be more than one way to formulate this puzzle, but a hint that may help you get started is to formulate the puzzle using three sets of decision variables: one representing vehicle moves, one representing positions occupied by vehicles and one representing the position occupied by the front of each vehicle." The students taking this course, primarily Masters degree students, were allowed to work in groups of three and were required to submit a written report showing both their formulation and solutions for the three assigned initial scenarios.

After completion of the project, students were asked informally about their experiences with the

Table 5 Solution to Example Rush Hour Puzzle

		Move vehicle marker	
Move #	Vehicle #	From position	To position
1	3	29	26
2	2	1	2
3	8	18	36
4	5	19	13
5	6	31	25
6	4	33	31
7	7	22	34
8	1	14	18

Rush Hour problem. In summary, all 11 students taking the course thought the Rush Hour problem was interesting, challenging, and a beneficial learning tool. Students said they spent more time on this project than on any of the other projects in the same course, which were comprised of more traditional operations research problems such as the assignment problem, vehicle routing, and network optimization. However, they reported enjoying the Rush Hour project the most because it was fun, and they "did not want to stop working on it until they got it right." The authors definitely noticed much more of a "buzz" of conversation surrounding this project than the other course projects, in class, in the computer lab, and during instructor office hours. Students felt this project improved both their formulation skills and their knowledge of the solution software.

7. Concluding Remarks

This paper indicates that games can be useful tools for teaching operations research techniques as a component of engineering design education. The literature suggests that learning can be greatly enhanced when students perceive their work to be fun. Games are particularly useful in stimulating creativity, because they allow students to apply design techniques in settings that may not be especially intuitive and because they assist in visualizing problem goals and solutions. Attempts to stimulate creativity through work on realistic state-of-the-art applications are also useful, but these applications are often beyond the ability of undergraduate students. Even so, games cannot be used exclusively to teach the desired techniques. As always, engineering graduates must have a practical bias and a thorough knowledge of the current practices of their chosen field. Games, such as the simple board puzzles discussed herein, can help students learn creative formulation and solution techniques and, therefore, can enhance their understanding of the foundations of solution techniques in more realistic settings.

References

Asao, M. 1990. Simulation game of experimental design. *Annual Quality Congress Trans.* **44** 881–886.

Baxter, N. 2006. http://www.puzzleworld.org/SlidingBlockPuzzles/rushhour.htm (last accessed on August 2, 2006).

Beasley, J. 1992. *The Ins and Outs of Peg Solitaire*. Oxford University Press, Oxford, England.

Berlekamp, E., J. Conway, R. Guy. 1982. *Winning Ways for Your Mathematical Plays*, Vol. 2: *Games in Particular*. Academic Press, London.

Bottomley, L. J., E. A. Parry. 1998. Exciting children about science and engineering: The science of playgrounds. *ASEE Annual Conf. Proc.* [CD-ROM], 5.

Carswell, L., D. Benyon. 1996. Adventure game approach to multimedia distance education. *Proc. 1996 Sympos. Integrating Tech. Comput. Sci. Ed.* ACM, Fort Collins, CO, 122–124.

Chlond, M. J. 2002a. The riddle of the pilgrims. *INFORMS Trans. Ed.* **2**(2), http://ite.pubs.informs.org/Vol2No2/Chlond/index.php (last accessed on August 2, 2006).

Chlond, M. J. 2002b. Unconstrained peg solitaire. *INFORMS Trans. Ed.* **2**(3), http://ite.pubs.informs.org/Vol2No3/Chlond/index.php (last accessed on August 2, 2006).

Chlond, M. J. 2002c. The traveling space telescope problem. *INFORMS Trans. Ed.* **3**(1), http://ite.pubs.informs.org/Vol3No1/Chlond/index.php (last accessed on August 2, 2006).

Chlond, M. J., O. Akyol. 2003. A nimatron. *INFORMS Trans. Ed.* **3**(3), http://ite.pubs.informs.org/Vol3No3/ChlondAkyol/index.php (last accessed on August 2, 2006).

Chlond, M. J., R. C. Daniel, S. Heipcke. 2003. Fiveleapers a-leaping. *INFORMS Trans. Ed.* **4**(1), http://ite.pubs.informs.org/Vol4No1/ChlondDanielHeipcke/index.php (last accessed on August 2, 2006).

Chlond, M. J., C. M. Toase. 2002. IP modeling of chessboard placements and related puzzles. *INFORMS Trans. Ed.* **2**(2), http://ite.pubs.informs.org/Vol2No2/ChlondToase/index.php (last accessed on August 2, 2006).

Chlond, M. J., C. M. Toase. 2003. IP modeling and the logical puzzles of raymond Smullyan. *INFORMS Trans. Ed.* **3**(3), http://ite.pubs.informs.org/Vol3No3/ChlondToase/index.php (last accessed on August 2, 2006).

Cowan, J. 1998. *On Becoming an Innovative University Teacher*. Open University Press, Philadelphia.

Cracker Barrel Old Country Store. 2006a. Peg game history. http://www.crackerbarrel.com/games-kids.cfm?doc_id=217 (last accessed on August 2, 2006).

Cracker Barrel Old Country Store. 2006b. Peg game. http://www.crackerbarrel.com/games-kids.cfm?doc_id=42 (last accessed on August 2, 2006).

Dym, C. L., A. M. Agogino, O. Eris, D. D. Frey, L. J. Leifer. 2005. Engineering design thinking, teaching, and learning. *J. Engrg. Ed.* **94**(1) 103–120.

Elliot Avedon Museum and Archive of Games. 2006. Peg solitaire. University of Waterloo, Ontario, Canada, http://www.gamesmuseum.uwaterloo.ca/VirtualExhibits/puzzles/solitare/index.html (last accessed on August 2, 2006).

Ermer, D. S., C. M. Phipps. 1996. Teamwork through tinkertoys and a successful egg drop. *Annual Quality Congress Transactions*. ASQC, Milwaukee, WI, 862.

Farkas, K. 1993. Informatics games for developing children's thinking ability. *IFIP Trans. A: Comput. Sci. Tech.* **A-34** 87–93.

Foulds, L., D. Johnson. 1984. An application of graph theory and integer programming: Chessboard non-attacking problems. *Math. Magazine* **57**(2) 95–104.

Gardner, M. 1969. Peg solitaire. *The Unexpected Hanging and Other Mathematical Diversions*. Simon and Schuster, New York.

Guy, R. 1998. Unsolved problems in combinatorial games. R. Nowakowski, ed. *Games of No Chance*. Cambridge University Press.

Heineke, J., L. Meile. 1996. Workshop on enhancing learning in the classroom using interactive games and exercises. *Proc. 27th Annual Meeting Decision Sci. Institute*, Vol. 1. 17.

Hill, J., C. Ray, J. Blair, C. Carver. 2003. Puzzles and games: Addressing different learning styles in teaching operating systems concepts. *SIGCSE Bull.* 182–186.

Jones, R. M. 2000. Design and implementation of computer games: A capstone course for undergraduate computer science education. *31st Tech. Sympos. Comput. Sci. Ed.* 260–264.

Kraitchik, M. 1942. *Mathematical Recreations*. Norton, New York.

Letavec, C., J. Ruggiero. 2002. The n-queens problem. *INFORMS Trans. Ed.* **2**(3), http://ite.pubs.informs.org/Vol2No3/LetavecRuggiero/index.php (last accessed on August 2, 2006).

Levitin, A., M. Papalaskari. 2002. Using puzzles in teaching algorithms. *SIGCSE Bull.* 292–296.

Mazeworks. 2006. Peg solitaire. http://www.mazeworks.com/peggy/ (last accessed on August 2, 2006).

Meyer, H. 2000. Fun for everyone. *IEEE Engrg. Management Rev.* **28**(2) 45–48.

Moore, C., D. Eppstein. 2002. One-dimensional peg solitaire and duotaire. R. Nowakowski, ed. *More Games of No Chance*. Cambridge University Press.

Nafalski, A., E. Milosz, M. Milosz. 1994. Software system for development and implementation of decision simulation games. *IEEE Internat. Conf. Multi-Media Engrg. Ed. Proc.* 65–71.

Pavlidis, T. 1997. Teaching graphics through video games. *Comput. Graphics* **31**(3) 56–57.

Rogers, C. B., S. Williams. 1993. Making it fun: learning and productivity. *Annual Quality Congress Trans.* **47** 536–541.

Schwarz, C. J. 2003. An online system for teaching the design (and analysis) of experiments. *2003 Proc. Amer. Statist. Assoc. Statist. Ed. Section*. [CD-ROM], American Statistical Association, Alexandria, VA.

Sniedovich, M. 2002a. OR/MS games: 1. A neglected educational resource. *INFORMS Trans. Ed.* **2**(3), http://ite.pubs.informs.org/Vol2No3/Sniedovich/index.php (last accessed on August 2, 2006).

Sniedovich, M. 2002b. OR/MS games: 2. Towers of Hanoi. *INFORMS Trans. Ed.* **3**(1), http://ite.pubs.informs.org/Vol3No1/Sniedovich/index.php (last accessed on August 2, 2006).

Sniedovich, M. 2003a. OR/MS games: 3. Counterfeit coin problem. *INFORMS Trans. Ed.* **3**(2), http://ite.pubs.informs.org/Vol3No2/Sniedovich/index.php (last accessed on August 2, 2006).

Sniedovich, M. 2003b. OR/MS games: 4. The joy of egg-dropping in braunschweig and Hong Kong. *INFORMS Trans. Ed.* **4**(1), http://ite.pubs.informs.org/Vol4No1/Sniedovich/index.php (last accessed on August 2, 2006).

ThinkFun, Inc. 2006. http://www.thinkfun.com/DEFAULT.ASPX?PageNo=HOME (last accessed on August 2, 2006).

Trick, M. 2001. Building a better game through dynamic programming: A flip analysis. *INFORMS Trans. Ed.* **2**(1), http://ite.pubs.informs.org/Vol2No1/Trick/Trick.php (last accessed on August 2, 2006).

Uehara, R., Iwata 1990. Generalized Hi-Q is NP-complete. *The TRANSACTIONS of the IEICE* **E73** 270–273.

Zafran, R. 1995. Laser applications in science education (LASE) games. *Proc. SPIE-The Internat. Soc. Optical Engrg.* 198–201.