Advance Access published on June 20, 2006
doi:10.1093/comjnl/bxl033

Solving Rummikub Problems by Integer Linear Programming

D. den Hertog[1]* and P. B. Hulshof[2]

[1]*Tilburg University, PO Box 90153, 5000 LE Tilburg, The Netherlands*
[2]*Dianapad 17, 5042 LM Tilburg, The Netherlands*
**Corresponding author: D.denHertog@uvt.nl*

The Rummikub problem of finding the maximal number or value of the tiles that can be placed from your rack onto the table is very difficult, since the number of possible combinations are enormous. We show that this problem can be modeled as an integer linear programming problem. In this way solutions can be found in 1 s. We extend the model such that unnecessary changes of the existing sets on the table are minimized.

Keywords: Rummikub, game, integer linear programming

Received 15 July 2005; revised 14 March 2006

1. INTRODUCTION

Computers have frequently been used to solve different kinds of puzzles or games. In many cases it appeared that the combination of computer power and efficient mathematical techniques is able to solve such problems. Wilson [1] used integer linear programming (ILP) for compiling crossword puzzles. Littman *et al*. [2] used a probabilistic approach to solve crossword puzzles. Iterative algorithms to solve the Hanoi or Reves puzzles are described by [3, 4, 5, 6, 7]. There are also many papers on optimal poker strategies, e.g. [8, 9]. A mathematical investigation of optimal algorithms for Mastermind is given in [10]. In *INFORMS Transactions on Education* several papers appeared on ILP formulations for several games and puzzles, e.g. The Riddle of the Pilgrims [11], Peg Solitaire [12], n-Queens problem [13], Einstein's riddle [14], Nim [15], Raymond Smullyan's puzzles [16], 2-egg puzzle [17], SuDoku and The Log Pile [18].

In this paper we will show how to solve combinatorial problems arising in Rummikub using ILP techniques. By using the combination of computer power and efficient mathematical techniques we are able to solve the Rummikub problems.

Rummikub is a well-known game for two to four players. The aim of the game is to be the first player who eliminates all the tiles from his rack by forming them into sets of runs and groups. You have to try to keep as few points on your rack as possible. Rummikub contains 106 tiles. There are two sets of tiles numbered from 1 to 13 in 4 colors: black, red, blue and orange. Furthermore, there are two joker tiles. There are two kinds of sets. The first one is a group. A group is a set of either three or four tiles of the same number but in different colors. The second is a run. A run is a set of three or more consecutive numbers, all in the same color. The jokers can be used for any tile in a set.

Every player takes 14 tiles. The remaining tiles on the table are the pool. Players must place sets valued at least 30 points (add up the numbers of the tiles in the sets) onto the table in the first move. This move is called the 'initial meld'. If unable to do an initial meld, or player chooses to delay initial meld, a tile must be taken from the pool and this concludes the player's turn. During the initial meld sets on the table may not be manipulated or built upon with tiles from player's rack. After players have made their initial plays, they can also manipulate sets on the table to combine them with tiles from their racks. Manipulation is the most exciting part of playing Rummikub. Players try to table the greatest amount of tiles, by rearranging or adding to sets that are already on the table. Sets can be manipulated in many ways as long as at the end of each round, only legitimate sets remain, and no loose tiles are left over. If a player cannot add onto the other sets, the player picks a tile from the pool and the turn ends. The round continues until one player empties his rack and calls 'Rummikub'. That player wins the game and the other players tally the numbers of the tiles they are holding on their racks. The joker has a penalty value of 30 points. The score is totaled as a negative amount. The winner receives a positive score equal to the total of all the losers' points. For a more detailed description of Rummikub we refer to the website http://www.rummikub.com.

One of the problems for a Rummikub player is to find the maximum number or value of the tiles you can place on the table, such that all Rummikub rules are obeyed. Since the number of possibilities can be astronomically high, simply trying all possibilities is impossible, even when the computer is used. Note that the number of possibilities can be extremely high at the initial play, since the '30 points' rule often leads to players acquiring large racks of tiles.

In this paper we describe an ILP model to solve this problem. For the theory of ILP, see [19] or [20]. Often there are many optimal solutions, i.e. the optimal number or value of tiles can be placed onto the table in many different ways. To save time for manipulating the existing sets on the table, one may look for the optimal solution with minimal changes. Hence, the main goal is still to maximize the number or value of the tiles that can be placed onto the table, but as a secondary goal the number of changes of the existing sets on the table is minimized. We describe how the model can be adjusted such that this can be accomplished. We give some examples to show that optimal solutions can be obtained very fast by using our model.

We emphasize that our model only optimizes the number or value of the tiles that can be placed on the table. For strategic reasons, a player can decide to place fewer tiles on the table. For example, sometimes it is useful to hold back the fourth tile of a group or run and lay only three, so that on the next turn one can lay a tile instead of drawing from the pool. However, for the player it is useful to know the maximal number or value of the tiles that can be placed onto the table.

2. ILP MODEL

Let us first count the number of possible different sets. The game contains 52 tiles of 4 different colors plus 2 jokers. Let us first consider sets without jokers. With every color you can make 11 different runs with 3 consecutive numbers, 10 runs with 4 consecutive numbers and 9 runs with 5 consecutive numbers. Note that a run with six or more consecutive numbers can be divided into runs of length three, four or five, and consequently need not to be considered separately. There are 13 numbers. With every number you can make four groups of three different colors and one group with four different colors. These numbers are summarized in the second column of Table 1.

Let us now consider sets containing exactly one joker. It is easy to check that with every color you can make 23 different runs with 3 consecutive numbers, 31 runs with 4 consecutive numbers and 37 runs with 5 consecutive numbers, where each run contains exactly one joker. There are 13 numbers. With every number you can make six groups of three different colors and four groups with four different colors, where each group contains exactly one joker. These numbers are summarized in the third column of Table 1.

TABLE 1. Numbers of possible sets

Run/group	Without joker	With 1 joker	With 2 jokers	Total
Three consecutive numbers	44	92	52	188
Four consecutive numbers	40	124	132	296
Five consecutive numbers	36	148	233	417
Same number/three different colors	52	78	0	130
Same number/four different colors	13	52	78	143
	185	494	495	1174

Let us now consider sets containing exactly two jokers. It is easy to check that with every color you can make 13 different runs with 3 consecutive numbers, 33 runs with 4 consecutive numbers and 58 runs with 5 consecutive numbers, where each run contains exactly two jokers. There are 13 numbers. With every number you can make six groups with four different colors, where each group contains exactly one joker. Note that groups with three different colors and two jokers are already counted as a run of three consecutive numbers! These numbers are summarized in the fourth column of Table 1.

This means that there are in total 1174 possible different sets. These possible sets play an important role in the model. We first give the model:

Indices

$i \in I$ type of the tile (defined by color and number), $I = \{1, 2, \ldots, 53\}$,

$j \in J$ number of set (run or group), $J = \{1, 2, \ldots, 1174\}$.

Parameters

s_{ij} indicates whether tile i is in set j (yes = 1, no = 0),

t_i tile i is 0, 1 or 2 times on the table,

r_i tile i is 0, 1 or 2 times on your rack.

Variables

x_j set j can be placed 0, 1 or 2 times onto the table,

y_i tile i can be placed 0, 1 or 2 from your rack onto the table.

Objective and constraints

$$\text{Max} \sum_{i=1}^{53} y_i$$

subject to

$$\sum_{j=1}^{1174} s_{ij} x_j = t_i + y_i \quad \forall i$$

$$y_i \le r_i \quad \forall i$$

$$x_j \in \{0,1,2\} \quad \forall j$$

$$y_i \in \{0,1,2\} \quad \forall i$$

The objective is the total number of tiles that can be placed onto the table. The first constraint ensures that you can only make sets of the tiles that are on your rack or on the table. The right-hand side of this constraint denotes the number of tile i that are already on the table plus that are placed from the rack onto the table. The left-hand side adds up the number of tile i present in the sets that are finally on the table. The second constraint states that the tiles you can place from your rack onto the table cannot be more than the tiles that are on your rack.

This model contains at most 1174 variables and 53 real constraints. Furthermore, note that this model is always feasible: $y_i = 0$ and objective value 0 corresponds with the current situation, i.e. no tiles are placed onto the table.

Instead of maximizing the total number of tiles, one can also maximize the total value of the tiles. Only the objective will change:

$$\text{Max} \sum_{i=1}^{53} v_i y_i$$

in which v_i is the value of tile i.

Now, the model is adjusted such that unnecessary changes of the existing sets on the table are avoided. The main goal is still to maximize the number or value of the tiles that can be placed onto the table, but as a secondary goal the number of changes of the existing sets is minimized. This is accomplished by the following model:

$$\text{Max} \sum_{i=1}^{53} v_i y_i + \frac{1}{M} \sum_{j=1}^{1174} z_j$$

subject to

$$\begin{aligned} \sum_{j=1}^{1174} s_{ij} x_j &= t_i + y_i \quad \forall i \\ y_i &\le r_i \quad \forall i \\ z_j &\le x_j \quad \forall j \\ z_j &\le w_j \quad \forall j \\ x_j &\in \{0,1,2\} \quad \forall j \\ y_j &\in \{0,1,2\} \quad \forall i \\ z_j &\in \{0,1,2\} \quad \forall j \end{aligned}$$

with the following new parameters:

w_j	set j is 0, 1 or 2 times on the table
M	constant (default value 40), and the following new variable:
z_j	set j occurs 0, 1 or 2 times in the old and in the new solutions.

The second term in the objective is the sum of all sets that were in the old situation and in the new situation, i.e. the sets that are kept unchanged. Consequently, this term will minimize the unnecessary changes. Observe that since we divide by M, the value of the second term is always less than one: there are 53 different tiles in the game and all of them are two times available and a set of tiles exist of at least three tiles, so there are atmost $35 = \lfloor (53 * 2)/3 \rfloor$ sets on the table. This means that a solution with a higher total tile value is always preferred, even if (many) more changes of the existing sets are necessary. Note that $z_j = \min(x_j, w_j)$ and therefore we have to add the two constraints $z_j \le x_j$ and $z_j \le w_j$. Moreover, observe that there are at most 35 relevant extra variables z_j, since there are atmost 35 sets on the table.

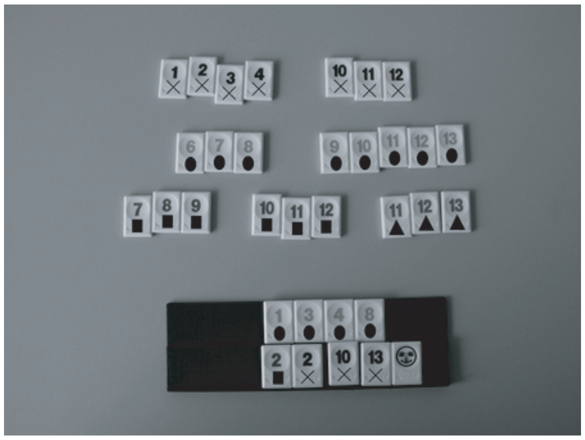
FIGURE 1. Starting position of example 1.

3. EXAMPLES

The ILP model is implemented in AIMMS (Advanced Integrated Multi-Dimensional Modelling Software, see [21]. We used the embedded XA-solver to solve the resulting ILP problems. The execution time (on a Pentium III) appears to be less than a second. We will show the results for two examples.

Example 1. This example will start with the following tiles, see Figure 1. In all examples, the black tiles are marked with a cross, the orange tiles with a circle, the blue tiles with a square and the red tiles with a triangle.

The solution when the objective is the number of tiles is given in Figure 2. The black 13 is added to the set ‘black 10, 11 and 12’ and the orange 8 is added before the orange 9. Furthermore, you have a joker and could put them together with the orange 1, 3 and 4 on the table.

The solution of Figure 3 is obtained by optimizing the total value of the tiles and minimizing the unnecessary changes on the tables. The joker is now used to put the black 10 onto the table. This is only one tile, but the value of this tile is higher than the three tiles you put onto the table in the previous solution. In Figure 3 you can see that the sets are very different to the sets in Figure 1. This illustrates that in our model the value of the tiles is more important than avoiding unnecessary changes as much as possible.

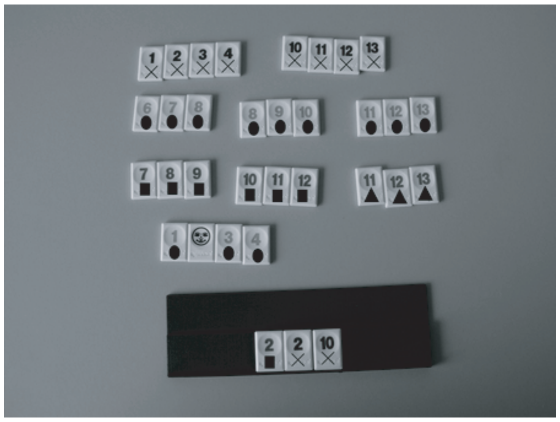

FIGURE 2. Solution of first model for example 1.

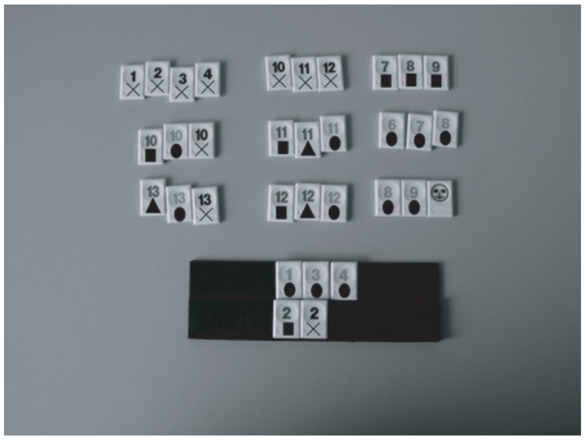

FIGURE 3. Solution of second model for example 1.

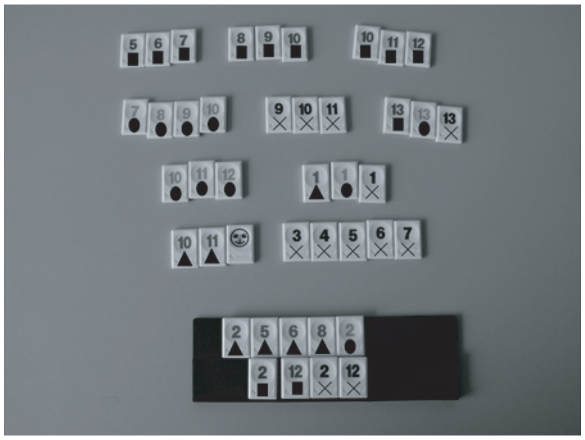

FIGURE 4. Starting position of example 2.

Example 2. In the second example the player can place all the tiles from his rack onto the table. The starting solution is given in Figure 4, and the solutions of the two models are given in Figures 5 and 6 respectively.

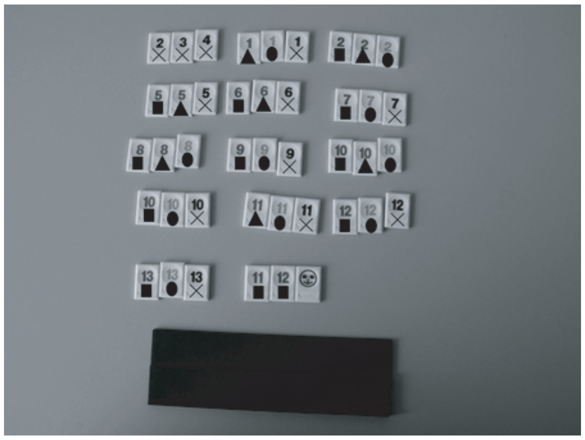

FIGURE 5. Solution of first model for example 2.

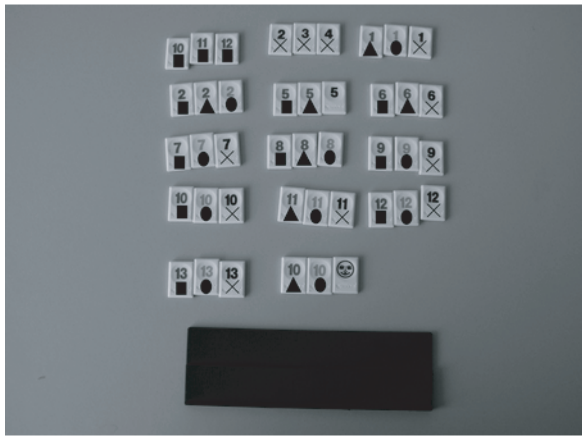

FIGURE 6. Solution of second model for example 2.

The difference between the two models is the formation of new sets of tiles. The first model results into a solution (Figure 5) in which two sets are the same as in the beginning of the play (the group with the ones and the thirteen-group). The solution of the second model (Figure 6) contains one extra set which is the same as in the starting position (the row blue 10, 11 and 12). The reason is of course that in the second model also the unnecessary changes are minimized.

ACKNOWLEDGEMENTS

The authors would like to thank the two anonymous referees for a number of useful comments that were incorporated into the finished article.

REFERENCES

[1] Wilson, J. M. (1989) Crossword compilation using integer programming. *Comput. J.*, **32**(3), 273–275.

[2] Littman, M. L., Keim, G. A. and Shazeer, N. (2002) A probabilistic approach to solving crossword puzzles. *Artif. Intell.*, **134**, 23–55.

[3] Gedeon, T. D. (1996) An iterative solution produced by transformation. *Comput. J.*, **39**(4), 353–356.

[4] Majumdar, A. A. K. (1994) A note on the iterative algorithm for the Reves puzzle. *Comput. J.*, **37**(5), 463–464.

[5] Sapir, A. (2004) The tower of Hanoi with forbidden moves. *Comput. J.*, **47**(1), 20–24.

[6] Vandeliefvoort, A. (1992) An iterative algorithm for the Reve puzzle. *Comput. J.*, **35**(1), 91–92.

[7] Sniedovich, M. (2002) OR/MS games: 2. Towers of Hanoi. *INFORMS Trans. Educ.*, **3**(1), 34–51.

[8] Cutler, W. H. (1975) An optimal strategy for pot limit poker. *Am. Math. Mon.*, **82**, 368–376.

[9] Cassidy, J. (1998) The last round of betting in Poker. *Am. Math. Mon.*, **105**(9), 825–831.

[10] Chen, S. T. and Lin, S. S. (2004) Optimal algorithms for $2 \times n$ Mastermind games—graph-partition approach. *Comput. J.*, **47**(5), 602–611.

[11] Chlond, M. J. (2002) The riddle of the pilgrims. *INFORMS Trans. Educ.*, **2**(2), 56–57.

[12] Chlond, M. J. (2002) Unconstrained Peg Solitaire. *INFORMS Trans. Educ.*, **2**(3), 99–100.

[13] Letavec, C. and Ruggiero, J. (2002) The n-Queens problem. *INFORMS Trans. Educ.*, **2**(3), 101–103.

[14] Yeomans, J. S. (2003) Solving 'Einstein's Riddle' using spreadsheet optimization. *INFORMS Trans. Educ.*, **3**(2), 55–63.

[15] Chlond, M. J. and Akyol, O. (2003) A Nimatron. *INFORMS Trans. Educ.*, **3**(3), 90–99.

[16] Chlond, M. J. and Toase, C. M. (2003) IP modeling and the logical puzzles of Raymond Smullyan. *INFORMS Trans. Educ.*, **3**(3), 1–12.

[17] Sniedovich, M. (2003) OR/MS games: 4. The joy of egg-dropping in Braunschweig and Hong Kong. *INFORMS Trans. Educ.*, **4**(1), 48–64.

[18] Chlond, M. J. (2005) Classroom exercises in IP modeling: Su Doku and the Log Pile. *INFORMS Trans. Educ.*, **5**(2).

[19] Schrijver, A. (1986) *Theory of Linear and Integer Programming*. Wiley, New York.

[20] Wolsey, L. A. (1998) *Integer Programming*. Wiley, New York.

[21] Bisschop, J. and Roelofs, M. (2003) *AIMMS: The User's Guide*. Paragon Decision Technology, Haarlem, The Netherlands.