Randomized Algorithms For Identifying Minimal Lottery Ticket Sets

FAYYAZ YOUNAS
Department of Computer Science, State University of New York at Stony Brook

(STEVEN SKIENA)
Department of Computer Science, State University of New York at Stony Brook

Abstract

This report presents the findings of a research project with Lotto Systems Group, Redwood City, CA. This research focuses on the question: "If a fortune-teller is able to provide a set of N numbers and guarantee that P of the future draws will be out of his/her set, how do we select a *minimum* set of tickets that will *guarantee* a win?" Initially a backtracking algorithm was implemented that systematically searches through the solution space of all possible ticket sets for a minimum set of tickets. Subsequent attempts were made to find faster algorithms for this exponential time problem using randomization. The solutions produced by these algorithms have been found to be close to optimal and better than published results.

Introduction

In a lottery, there is a set of M possible numbers to choose from. For example, in the New York Pick-6 Lotto, M is equal to 54. The lottery is defined by two other parameters: the number of numbers picked per ticket, R (in the New York Pick-6 Lotto, R = 6) and the minimum number of correct numbers on a ticket to win a prize, J. Here we assume a clairvoyant fortune-teller who has narrowed the set of numbers to choose from, promising that P numbers will be drawn from a predicted set of size N. With this insight, we seek to buy the smallest number of tickets to guarantee a win.

In order that a *minimum winning* set be theoretically computable, we assume that the following input constraints are met: J <= P <= R <= N. Notice in particular that P should be greater than or equal to J, or in other words, the fortune-teller must guarantee, from his/her set, at least the minimum number of outcomes required to win a prize in a future draw.

The problem of finding a *minimum* set of tickets that will *guarantee* a win is not a trivial one. Given that P out of R outcomes will be from the fortune-teller set, it is not difficult to see that there are ${}^{N}C_{P}$ = (N!/P!)/(N-P)! possible P-subsets from the fortune-teller set that can occur in the winning ticket. If we were to pick all P-subsets from the fortune-teller set W times and fill in the remaining R-P slots arbitrarily, the set of tickets obtained will have at least W occurrences of each P-subset and guarantee us W wins. However, such a set need not be a minimal one and in most cases is not.

We know from the fortune-teller's promise that one of the P-subsets will occur in the winning ticket. It is possible for two P-subsets to differ by less than J numbers. When such a situation arises, the subsets are said to *overlap* or *cover* one another with respect to the shared J numbers and only one of the P-subsets must be in a purchased ticket. This phenomenon is best illustrated using an example. Suppose we are playing the PICK-4 Lotto and require one 2/4 win. Hence R=4, J=2 and W=1. Furthermore let's assume that the fortune-teller predicts 3 numbers from a set of 5 numbers (i.e. P=3 and N=5). If all P-subsets were taken from the fortune-teller set and arbitrarily filled to complete the tickets, we would have a set of ten tickets that guarantees one 2/4 win (See Figure 1). However, it is also possible to exclude some tickets from this set because of several two-number overlaps. For instance the subset {3, 4, 5} is different than {1, 3, 5} by only one number and it will be wasteful to use both of these in purchased tickets. We might think that not including {3, 4, 5} will permit the possibility of losing, but that is not the case since if {3, 4, 5}occurs we will have '3' and '5' in {1, 3, 5}that we bought to claim the prize! Similarly there can be many more redundant P-subsets. An optimal solution is shown in Figure 2. Our lottery problem is that of finding the smallest set of P-subsets from the fortune-teller set that guarantees the specified number of wins by keeping the number of overlaps to a minimum. This set of P-subsets defines the winning set regardless of what numbers are used to complete the R slots on the ticket.

In order to obtain an estimate for the optimal cardinality of the winning ticket set, we define a distance function that maps two P-subsets to an integer equal to the number of different members in the two sets. With the given input constraint, each P-subset covers or overlaps with other P-subsets at distances less than or equal to P-J. In other words, if the distance between two P-subsets is equal to P-J or greater, the two subsets do not overlap. The total number of other subsets covered by a particular P-subset is given by the sum of those at a distances from 0 to P-J. As explained earlier a naive way to guarantee W wins is to pick $W \times {}^{N}C_{P}$ P-subsets. Such a set is not optimal because of overlaps. Each P-subset overlaps with $\sum_{I} {}^{N-P}C_{I} * {}^{P}C_{I}$ subsets and so assuming all overlaps can be avoided, the lower bound would be $(W \times {}^{N}C_{P}) \div \sum_{I} {}^{N-P}C_{I} * {}^{P}C_{I}$. However, it should be noted that this lower bound is not always obtainable because the assumption will not always be true.

Each of these P-subsets could either form a ticket in the winning ticket set or be left out (i.e. there are two choices). Hence, the total number of possible ticket sets is given by the formula 2^{λ}, where $\lambda = {}^{N}C_{P}$. To find an exact solution to the problem, each of these ticket sets will have to be examined to check if it guarantees a win or not; the smallest of these sets would then be picked. The checking process itself does not have a polynomial time complexity. There are ${}^{N}C_{J}$ possible subsets in the fortune-set which are equally likely to occur and a winning ticket set must include all of these. Hence the total time complexity of an exact algorithm is $2^{\lambda} * {}^{N}C_{J}$. The lottery problem requires exponential time to solve exactly.[1] Hence a backtracking algorithm which attempts to solve the lottery problem exactly, will inevitably take ages to compute when encountered with a difficult set of input data (For example, when λ is large).

Добавлено примечание ([AJ1]):

Objective

The objective is to use heuristics such as randomization to develop approximate algorithms to solve the lottery problem that are much faster than the exact backtracking algorithm and yet produce winning ticket sets with cardinality close to the lower bound.

Procedure

Backtracking

The first algorithm implemented is an exact backtracking algorithm similar in nature to the algorithm used to solve the Eight Queens Problem[2]. A naive backtracking algorithm, simply searches through the solution space of all possible ticket sets and keeps record of the minimum winning set. Such an algorithm is doomed to hit a combinatorial wall for even modest size input parameters. An improvement to the naive algorithm is to incorporate 'pruning' of the solution space to be explored. In other words, not every ticket set need be explored. In spite of employing pruning strategies, this algorithm is bound to be slow and hence will not run to completion for difficult input data.

Добавлено примечание ([AJ2]):

Randomization

The two randomized algorithms implemented use a linear congruential random number generator to produce random tickets.[3] These are variants of the same basic algorithm which is to select a group of P-subsets from the fortune-teller set that guarantee the type of win required W times with minimum overlaps. The selected P-subsets then define a winning set regardless of the numbers picked to complete the tickets. The algorithms sample β random P-subsets at a time and each time pick the one that covers or overlaps with the most uncovered subsets of the fortune-teller set. In doing so they also keep a count of how many times each subset was covered. The algorithms are outlined below for the interested reader.

Algorithm #1

A. For i = 1 to W do:
    1. Randomly pick 'total' P-subsets one at a time and add a subset iff the # of overlaps with another picked subset is < i
B. While the ticket set does not guarantee W wins do:
    1. Sample β random P-subsets.
    2. Add the subset with maximum win coverage.

Algorithm #2

A. While the ticket set does not fulfill required wins, do:
    1. Sample β random P-subsets.
    2. Add the subset with maximum win coverage.

Results

Results on various input data show that, given enough time, a backtracking algorithm will generate winning ticket sets with cardinality equal to or slightly greater than the lower bound, because our lower bound may in fact not be achievable. However, it is only practicable to use such an algorithm if the total number of tickets sets (i.e. λ) is small since it is an exponential time algorithm. Unfortunately, λ is typically too large for input parameters of practical interest and so this approach ends up with winning ticket sets of cardinality well above the lower bound when constrained by time.

Experiments with the randomized algorithms show that the performance increases as the sampling pool, β, is increased from 1, but becomes steady for β greater than 100 (See Figure 3). This suggests that the algorithms do in fact approach the optimal obtainable ticket set. Also, performance is not increased if 2 or 3 tickets are collectively sampled at a time instead of only 1 ticket at a time. These algorithms are much faster than the exact algorithm and produce results close to the lower bound. Some solutions of general interest are shown in Table 1. A comparison with published solutions[4] shows better performance in some cases (see Table 2). For example, in Pick-5 Lotto, the published solution offers 9 tickets for 7.14 % win probability. Our algorithms provide similar coverage per number of tickets bought while achieving a100% win probability, which is inherently more difficult to do.

An interesting observation is that the number of tickets required for W wins is not just W times the number of tickets required for one win (see Table 1). The reason for this is that even though the lower bound calculated above assumes no overlap it is almost impossible to have no overlaps. However, if these overlaps are recorded they can be used towards another win, hence reducing the number of extra tickets required. It is also interesting to note that as the number of subsets increases, the greater are the chances of overlap and the deviation from the lower bound (see Figures 4A and B).

Conclusions

From the research conducted on the lottery problem it can be concluded that approximate methods using heuristics such as randomization can be devised, to produce close to optimal solutions in short time intervals. Our work has been licensed by Applied Lotto Systems for use in their future products.

References

[1] Cormen, Leiserson, and Rivest, *Introduction to Algorithms*, MIT Press, 1990.
[2] Tucker, *Applied Combinatorics*, John Wiley & Sons, 1984.

[3] Nijenhuis and Wilf, *Combinatorial Algorithms*, Academic Press, 1975.
[4] Serotic R., *Pick-5 Lotto*, L.S.I. Publishing, Inc., Menlo Park, CA, 1989.

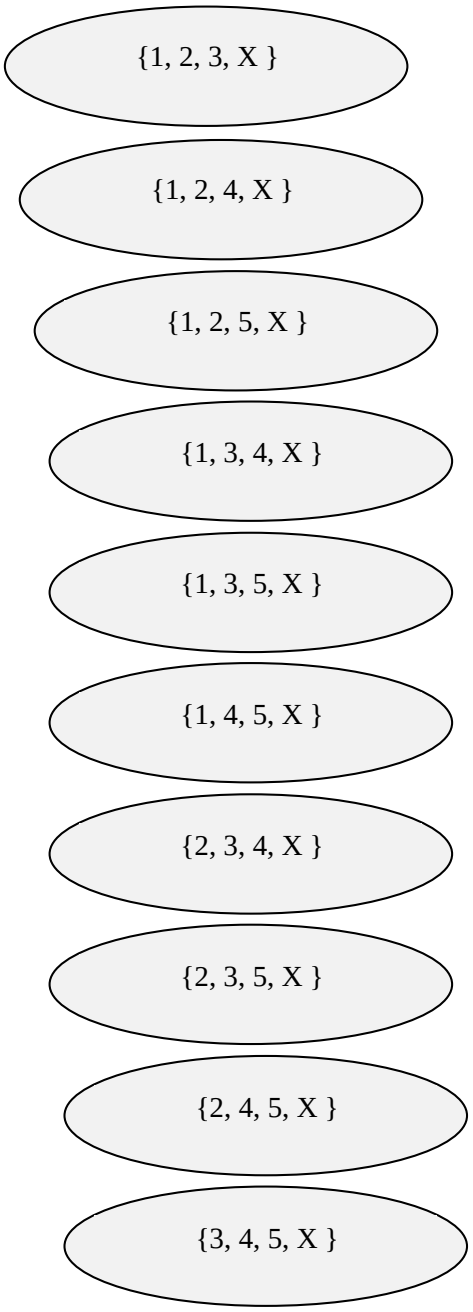

FIGURE 1
All 3-subsets of the 5 numbers in the fortune-teller set are used to form ten tickets that will guarantee one 2/4 win because one of these subsets has been promised to occur in the winning ticket. An "X" represents an arbitrarily picked number to complete the ticket.

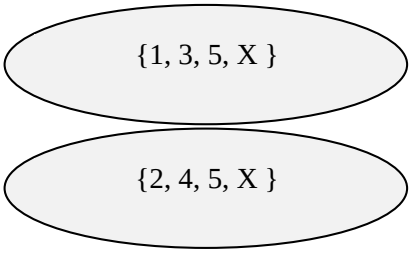

FIGURE 2
Only two of the ten 3-subsets of the 5 numbers in the fortune-teller set are used to form a minimal ticket set that will guarantee one 2/4 win. This is possible because there are many two number *overlaps* between the 3-subsets. An "X" represents an arbitrarily picked number to complete the ticket.

Numbers to be picked in a game.	Numbers required for a win.	Size of fortune-teller set.	Numbers promised from the FT set.	Number of wins required.	Estimated lower bound.	Number of tickets identified.	Time / sec IBM 486 DX w/ Linux OS
5	4	15	5	1	58	137	95
5	4	15	5	2	117	218	147
5	4	15	5	3	176	294	163
5	5	15	5	1	3003	3127	333
6	4	15	5	1	58	138	145
6	5	15	5	1	3003	3109	346
6	6	15	6	1	5005	5129	503
5	4	18	5	1	129	330	432
6	4	18	5	1	129	330	449
10	6	18	7	1	408	1080	919

TABLE 1

Source.	*Size of fortune-teller set.*	*Numbers promised from FT set.*	*Number of tickets identified.*	*Win probability.*	*Number of tickets for 1% coverage.*
Reference 4	9	5	9	7.14 %	1.26
Algorithm 2	9	5	127	100 %	1.27
Reference 4	10	5	14	5.56 %	2.52
Algorithm 2	10	5	254	100 %	2.54
Reference 4	11	5	11	2.38 %	4.62
Algorithm 2	11	5	467	100 %	4.67
Reference 4	13	5	57	4.42 %	12.89
Algorithm 2	13	5	1351	100 %	13.51

TABLE 2
This table compares the results obtained with Algorithm #2 to those published in reference 4. The results are for PICK-5 Lotto (i.e. R = 5) and one required win for the jackpot (i.e. J = 5 & W = 1).

[1]Cormen, Leiserson, and Rivest, *Introduction to Algorithms*, MIT Press, 1990.
[2]Tucker, *Applied Combinatorics*, John Wiley & Sons, 1984.
[3]Nijenhuis and Wilf, *Combinatorial Algorithms*, Academic Press, 1975.
[4]Serotic R., *Pick-5 Lotto*, L.S.I. Publishing, Inc.