Redback Operations

Algo Team
Shashvat Joshi

Research ML Algorithms to improve Efficiency in incentives Problems

Research

Incentive allocation is a common problem in incentive allocation system. For our gaming system Each activity that user does will produce Coins, the number of coins gathered through that activity will vary according to the effort of the user and on the kind of activity itself (one game might produce more coins than the other because it involves more physical effort).

Web development team also intends to count steps of the users who are walking/running through the mobile application it is building and will hand out coins based on the number of steps. This will also include levels.

Such allocation problems can be easily formulated as integer programming problems, which can be further relaxed to be linear. Although such problems have simple structures, their problem size can easily grow large (e.g., there can be many incentives, or a large population of gamers). To improve the efficiency of such large system we take help from an algorithmic approach as described below.

Problem Solution

- There is a total of m riders. –
- For each rider i = 1, m, there are k_i distinct types of coins that might be assigned to that rider. Note that k_i can be different for different riders.
- For each rider i and coin type j, there is an associated positive cost c_{ij} and an associated positive value v_{ij} where i = 1, ……, m; and j = 1, ……, k_i.
- Each rider is assigned at most one coin.
- There is a budget C, that limits the total cost that may be incurred in a feasible assignment.
- We define x_{ij}, where i = 1, ……, m; and j = 1,……, k_i, as a binary variable indicating whether passenger i is assigned a coin of type j .
- The aim is to maximize the total value of the coins assigned.

We can replace riders with gamers.

The pseudo code for solving the dual LP to get the primal coin allocation looks like the following:

pre-process: shortlist coins such that the costs, values and efficiencies of feasible coins for each rider are in increasing order; further shortlist coins and calculate breakpoints for each rider based on (E).

prepare breakpoints: for each rider, maintain a priority queue of coins and their associated costs, values and breakpoints (based on the breakpoints from largest to smallest, which has the nice property of being the same as ranking costs or values from smallest to largest); we have m queues where each item in a queue looks like (breakpoint, coin, value, cost, rider index).

initialize: set slope S = C (budget); and an active list L that includes m breakpoints, each being the item with the largest breakpoint from each of the m queues.

while S > 0 and length(L) > 0 **do**
breakpoint = pop largest from L (we know which rider this breakpoint refers to);
update S = S + cost from last breakpoint of this rider - cost from breakpoint of this rider;

> **if** there are remaining coins in the breakpoint rider's queue then next breakpoint = pop largest from the breakpoint rider's queue; push next breakpoint to L;
> **end**

end

allocate: once the while loop is done, we find the last breakpoint right before the one in the final active list L (if available) for each rider, and assign the corresponding coin for this last breakpoint; riders without available last breakpoints do not receive coins.

We are going to get at most one variable with a fractional solution in this relaxation, which is acceptable in exchange for the efficiency of solving this problem:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{m} \sum_{j=1}^{k_i} v_{ij} x_{ij} \\
\text{subject to} \quad & \sum_{j=1}^{k_i} x_{ij} \leq 1, \text{ for each } i = 1, \dots, m; \\
& \sum_{i=1}^{m} \sum_{j=1}^{k_i} c_{ij} x_{ij} \leq C; \\
& x_{ij} \geq 0, \text{ for each } i = 1, \dots, m, j = 1, \dots, k_i
\end{aligned}
$$

Pre-processing shows

$$0 < v_{i1} < v_{i2} < \cdots < v_{ik_i}, 0 < c_{i1} < c_{i2} < \cdots < c_{ik_i}, \text{ and } 0 < \frac{c_{i1}}{v_{i1}} < \frac{c_{i2}}{v_{i2}} < \cdots < \frac{c_{ik_i}}{v_{ik_i}}$$

Coins eliminated in the pre-processing step are dominated by at least one coin in the feasible set. In other words, the cost, value and efficiency of feasible coin for each given gamer are in increasing order.

We define the following dual variables:

Let y_i, i = 1..., m, be the dual variable corresponding to the constraint bounding each rider i to at most 1 coin.

Let λ, be the dual variable corresponding to the budget constraint.

Then the dual (D) is as follows:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{m} y_i + C\lambda \\ \text{subject to} & y_i + c_{ij}\lambda \geq v_{ij}, \text{ for each } i = 1, \dots, m, j = 1, \dots, k_i \\ & y_i \geq 0, \text{ for each } i = 1, \dots, m \\ & \lambda \geq 0. \end{array}$$

Let D(λ) denote the dual linear program (D) for a fixed value of λ ≥ 0. In considering this LP, note that the optimization problems for each rider i = 1, ….., m, is an independent (trivial) problem: we simply set

$$y_i^*(\lambda) = max\left\{0, \max_{j=1,\dots,k_i} \left(v_{ij} - \lambda c_{ij}\right)\right\}$$

and hence the optimal value to D(λ) is

$$\lambda C + \sum_{i=1}^{m} max\left\{0, \max_{j=1,\dots,k_i} \left(v_{ij} - \lambda c_{ij}\right)\right\}$$

We denote this expression as (E) for the rest of the post and minimize λ to solve (D), that minimizes this expression (E).

Solution of Dual LP

The below algorithm shows the compact form of the dual given as finding λ to minimize (E) provides a direct way to solve this LP.

Let's focus on one gamer i and the corresponding term in (E). We observe the following:

- Consider the two lines $v_{ij} - \lambda c_{ij}$ and $v_{ik} - \lambda c_{ik}$ for $j < k$. The two lines intersect at $\hat{\lambda} = \frac{v_{ik} - v_{ij}}{c_{ik} - c_{ij}}$
- Given our sorting properties of the data, we know that both the numerator and the denominator are positive, and max $\{v_{ij} - \lambda c_{ij}, v_{ik} - \lambda c_{ik}\}$ corresponds to option k in the interval [0, $\hat{\lambda}$] and corresponds to option j in the interval $\left[\hat{\lambda}, \frac{v_{ij}}{c_{ij}}\right)$
  (For larger $\hat{\lambda}$, we are in the case that both terms ez, are nonpositive and hence contribute 0 to (E).)

Therefore, we want to create a list of breakpoints for gamer i by checking each of the intersections of the k_i lines, starting with the intersection of j =1 and j =2. The key point here is that we only keep breakpoints that are monotonically decreasing. After this process, we may assume without loss of generality that each rider i, who is now left with g_i ($g_i \leq k_i$) incentives, has a list of eligible coins and breakpoints that satisfy

$$\lambda_{i1} > \lambda_{i2} > \lambda_{i3} > \cdots > \lambda_{ig_i}, \text{ where } \lambda_{i1} = \frac{v_{i1}}{c_{i1}} \text{ and } \lambda_{ij} = \frac{v_{ij} - v_{i,j-1}}{c_{ij} - c_{i,j-1}} \text{ for } j = 2,3, \ldots, g_i;$$

$$0 < v_{i1} < v_{i2} < \cdots < v_{ig_i}, 0 < c_{i1} < c_{i2} < \cdots < c_{ig_i}, \text{ and } 0 < \frac{c_{i1}}{v_{i1}} < \frac{c_{i2}}{v_{i2}} < \cdots < \frac{c_{ig_i}}{v_{ig_i}}.$$

The figure below provides a graphical explanation.

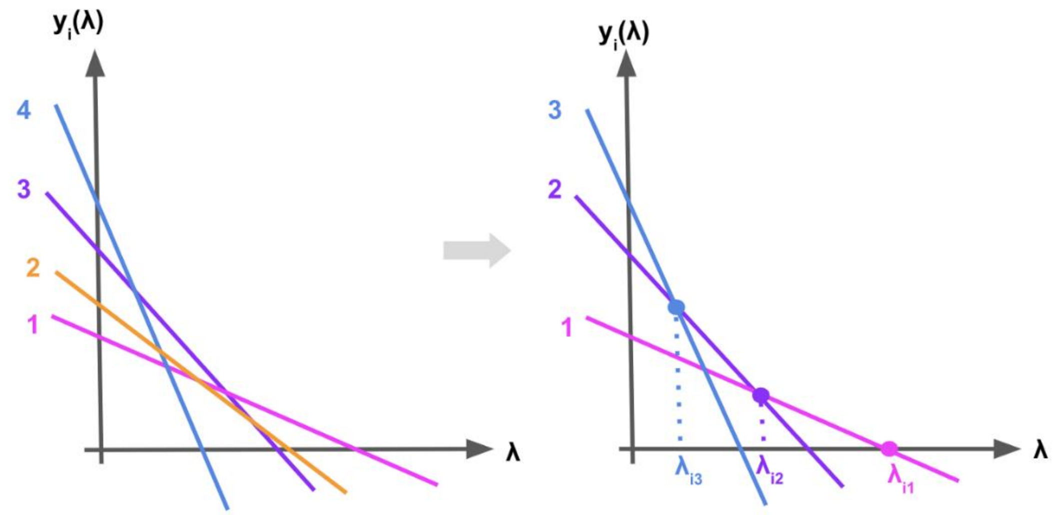

***Fig.**: Graphical illustration of the elimination and breakpoint creation step.*

Given a gamer who starts with 4 eligible coins indexed by 1, 2, 3 and 4, we first draw the lines $v_{ij} - \lambda c_{ij}$ (j = 1, 2, 3, 4) to construct the function y_i *(λ).* Coin 2 is then eliminated because its line does not contribute to the frontier of y_i *(λ)*, which means that the constraint corresponding to it is never tight at optimality, i.e., coin 2 will never be chosen. We are left with coins 1, 3, and 4 in our eligible set, and we reindex them to be coins 1, 2 and 3 for simplicity. We have three breakpoints λ_{i1}, λ_{i2} and λ_{i3}, where coin 1 is chosen for in λ in [λ_{i2}, λ_{i1}], coin 2 is chosen for in λ in [λ_{i3}, λ_{i2}] and coin 3 is chosen for in λ in [0, λ_{i3}]. In this example, k_i =4 and g_i = 3 after the elimination process.

Now consider what we have learned about the objective function (E). For a fixed rider i, the corresponding term in the summation gives rise to a piecewise linear function with the g_i breakpoints. Hence, in considering the sum in (E), we again get a piecewise linear function with the union of these breakpoint values (over all choices of i) as the resulting breakpoints. Let $0 = \rho_0 < \rho_1 < \cdots < \rho n$ be the union of these breakpoints. Between any pair of breakpoints, the function D(λ) is an affine function (i.e., it is of the form αλ + β); furthermore, D(λ) is convex with the slopes in these pieces are an increasing function of λ. For λ sufficiently large (bigger than ρn), the slope is clearly C; at λ = 0, the slope is

$$C - \sum_{i=1}^{m} c_{ig_i}$$

which we can assume is negative. Otherwise, we would simply assign each rider their maximum value coin (which is also their maximum cost coin), since that does not exhaust the budget, and hence is clearly optimal. In all other cases, there is a breakpoint ρ* for which the slope for λ < ρ* is negative, and for λ > ρ* is positive. In that case, D(λ) is minimized when λ = ρ*.

From optimal dual to optimal primal LP solution

Let's try optimal dual solution. We have already assumed that the slope of D(λ) at λ = 0 is negative, and hence λ* > 0. To compute the optimal primal solution, we do use the complementary slackness conditions for primal and dual LP optima, which gives us the following:

Since λ* > 0, we know that the budget constraint in the primal must hold with equality, i.e., $\sum_{i=1}^{m} \sum_{j=1}^{k_i} c_{ij} x_{ij} = C$

Similarly, we know that if x_{ij} > 0, then the corresponding dual constraint must hold with equality, i.e., $y_i + c_{ij}\lambda = v_{ij}$. In other words, for the optimal λ*, the corresponding *j** for rider i is an option for which the maximum is attained.

The case in which each breakpoint corresponds to a distinct unique rider is easy to understand. In this case, the optimal value λ* for (D) is the breakpoint corresponding to exactly one rider i*, and is between breakpoints for every other rider. Thus, for all other riders other than rider i*, there is a unique maximum j_i attained in the term corresponding to i in the sum (E); hence, the corresponding option must be set equal to 1 for that rider. This argument can be extended to the case when the optimal breakpoint corresponds to multiple riders. The correctness of the algorithm follows directly from showing this pair of primal and dual linear programming solutions are both feasible for their respective optimization problem, and satisfy the complementary slackness conditions.

References

1. Renner Philip, Scheidegger Simon, (2018, January, 18), *"Machine learning for dynamic incentive problems"*
   [Website] - https://economics.yale.edu/sites/default/files/jobmarketpaper_scheidegger_2.pdf

2. Wang Shujing, (2019, Sept, 4), "*Linear Optimization problem on incentive allocation"*
   [Website] - https://eng.lyft.com/how-to-solve-a-linear-optimization-problem-on-incentive-allocation-5a8fb5d04db1