How to select the best set of ads: Can we do better than Greedy Algorithm?

Xinle Liu

Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043

Selecting the best set of ads is critical for advertisers for a given set of keywords, which involves the composition of ads from millions of candidates. While click through rates (CTRs) are important, there could be high correlation among different ads, therefore the set of ads with top CTRs does not necessarily maximize the number of clicks. Greedy algorithm has been a standard and straightforward way to find out a decent enough solution, however, it is not guaranteed to be the global optimum. In fact, it proves not to be the global optimum more than 70% of the time across all our simulations, implying that it’s very likely to be trapped at a local optimum. In this paper, we propose a Greedy-Power Algorithm to find out the best set of creatives, that is starting with the solution from the conventional Greedy Algorithm, one can perform another Greedy Algorithm search on top of it, with the option of a few or even infinite rounds. The Greedy-Power algorithm is guaranteed to be not worse, as it only moves in the direction to increase the goal function. We show that Greedy-Power Algorithm’s performance is consistently better, and reach the conclusion that it is able to perform better than the Greedy Algorithm systematically.

I. INTRODUCTION

Ads has been contributing to more than 90% of revenue for companies as Google, Facebook, etc., therefore improving the performance deserves lots of time and efforts, not only for advertiser themselves, but also for various platform providers as Google. While advertisers have been spending lots of efforts generating ads with higher quality, it is never satisfying enough with multiple reasons. It’s subtle to find out the difference of two ads, especially when the similarity is high, humans can hardly find out the minor performance difference, especially without a thorough understanding of users, even when they’re expressing strong interests in their products. Also it suffers from the notorious scaling issue, since ads human writers’ efforts do not scale up as machines in the modern era of big data.

II. METHODOLOGY

A. Formulation of the problem

For each ads group, advertisers could specify a set of keywords to be matched:

\[ w \equiv (\bullet \ w_1 \ \bullet \cdots), i = 0, 1, \cdots, W - 1 \]  (1)

and provide a set of creatives:

\[ c \equiv (\bullet \ c_j \ \bullet \cdots), c = 0, 1, \cdots, N - 1 \]  (2)

With those two as the two dimensions we define the clicks matrix as below:

\[ K \equiv \begin{pmatrix} \cdots & \cdots & \cdots \ \cdots & K_{ij} & \cdots \ \cdots & \cdots \cdots & \cdots \end{pmatrix} \]  (3)

Similarly we could define the pCTR matrix.

Given the limit on number of creatives for each ads group, there could only be \( M \) creatives to be the final candidates, i.e. it is to select a subset of \( M \) elements from \( c \):

\[ d = (\cdots d_a \cdots), a = 0, 1, \cdots, M - 1 \]  (4)

to maximize the number of clicks, with goal function:

\[ G(d) = \sum_i \max\{K_{ia}, a = 0 \cdots M - 1\} \]  (5)

The shorthand notation of Greedy Algorithm \( G(W, N, M) \) and exact solution of \( E(W, N, M) \) will be used through this paper. The main assumption is that for each keyword request from external user, there could be only one creative to be selected, and we require it to be the one with the largest pCTR, so as to maximize revenue.

B. Greedy Algorithm

Greedy Algorithm has been a popular approach for many practical problems, in some cases it proves to be the optimal solution, while in most cases it can only provide a local optimal solution, though it is decent enough and could be a very good approximation to the exact optimal solution.

Greedy Algorithm with size \( M \) is similar to mathematical induction, starting with the same problem with smallest size, normally 0 or 1, one can go one step further for each iteration, and after \( M \) steps it solves the original problem, though not guaranteed to be globally optimal.

Greedy Algorithm for our problem works as the following:

1. \( a = 1 \): select creative \( \text{argmax}_j \sum_i K_{ij}, j \in [0, N - 1] \), i.e. the column with the largest sum over all keywords;

*lxinle@sas.upenn.edu, liuxl@google.com*
2. \ldots
3. \( a = a: \) Select the creative \( \text{argmax}_j G(d \cup \{c_j, j \in [0, N-1], c_j \notin d\}) \);
4. \ldots
5. Until \( M \) creatives are selected.

C. Greedy-Power Algorithm: \( G^n(r, f; W, N, M) \)

Assuming that Greedy Algorithm is a decent approximation to the exact solution \( \tilde{d} \), it is likely that the difference between those two sets are small, i.e. the number of creatives in the difference set is \( m \), which is much smaller than \( M \). In other words, starting with a solution from Greedy Algorithm, one only needs to perform a minor tuning and to replace a few creatives if ever necessary, when greedy algorithm is different from that exact solution \( \tilde{d} \). Here we introduce three parameters, \( r \) as the number of creatives removed from the original solution, \( f \) as the number of creative subsets as starting points, and \( n \) as the number of iterations on top of existing solution.

1. With \( r \) creatives removed from the original Greedy solution, one gets a new starting point and can perform another round of greedy algorithm with problem size \( M - r \to M \). It has the same complexity to the original Greedy Algorithm with a factor of \( \frac{r}{M} < 1 \). Note that the two solutions can be different by at most \( r \) creatives.
2. Removing \( r = 1, 2, 3 \ldots \) creative has \( M, M(M - 1)/2, M(M-1)(M-2)/6 \ldots \) options respectively. When \( r = 1 \), the cost is essentially the same to original Greedy Algorithm run with \( M \) iterations, since here it is \( M \) runs with \( 1 \) single iteration for each run. When \( r > 1 \), there could be a few options:
   - Systematic approach to enumerate all combinations, with cost increasingly quickly with \( r \), and much more expensive than the original Greedy Algorithm solution;
   - By sampling \( f \) unique combinations from all available options, one can keep the cost comparable to original one.

The \( r > 1 \) option could be better than \( r = 1 \), since it has the ability to remove multiple elements together, while the latter option can at most replace 1 creative. A simple calculation for a solution with elements \( c_x \) and \( c_y \) and an even cut, i.e. \( r = M/2 \) and \( M \) is even, the probability of \( c_x \) and \( c_y \) not being in the same half is:

\[
P(\{c_x\}, \{c_y\}) = \frac{C_{M-2}^{r-1}}{C_{M-1}^r} = \frac{M-r}{M-1} \sim \frac{1}{2}
\]  

At the same time, the probability of \( c_x \) and \( c_y \) in the same half is also roughly \( \frac{1}{2} \), so that they could be simultaneously removed when necessary. Therefore, a larger \( r \) has much more flexibility than a smaller \( r \).

\( n: \) With \( r \) creatives replaced by the 2nd round of Greedy Algorithm, one can ask the same question again, can we do better than the current solution? Actually the question is exactly the same to what we have been asking with a Greedy Algorithm in the first step. In fact, the new solution is from Greedy Algorithm as well, and essentially no different from its counterpart in the first round. As long as the goal function keeps increasing, one can continue this process, until goal function is the same to the previous iteration, which implies that we’re not able to go further, and it is equivalent to \( n = \infty \) in this case.

With those 3 parameters, we’d use the notation \( G^n(r, f; W, N, M) \) for our proposed Greedy-Power Algorithm.

III. SIMULATION

With the Greedy-Power Algorithm \( G^n(r, f; W, N, M) \), we’re able to run some simulations comparing against its baseline Greedy Algorithm.

A. Simulation Setup

- Matrix elements are generated from normal distribution and then take their absolute value;
- Matrix sizes are at the order of \( 50 \times 500 \) with \( M <= 10 \);
- By default, all simulations are run multiple times with trajectories \( T = 500 \) for each simulation, repeating 3 times.

B. Simulation Results and Discussion

1. \( G^2(r, \cdot \cdot \cdot ) \)

Note that as \( r \) increases, the computational cost increases by a factor of \( r \), with \( r \) iterations compared with one single iteration when \( r = 1 \). Strictly speaking, we are supposed to compare performance of \( G^n(1, M; W, N, M) \) vs \( G^n(r, M/r; W, N, M) \), since the computational cost match. However, the search space size increases exponentially with \( r \), while a factor of \( \frac{1}{r} \) would effectively remove more options, resulting in a quickly decreasing coverage ratio in the search space. Therefore, we’d remove the \( \frac{1}{r} \) factor and keep the default value \( f = M \) unless otherwise noted, i.e. run same number of creative subsets from the starting solution obtained from the Greedy Algorithm.

From those \( G^2(r; 30, 300, 6) \equiv G^2(r; 6; 30, 300, 6) \) simulation data with \( r = 1, 2, 3 \) in Tables I, II and III, one can clearly see that as \( r \) increases, the likelihood to go out of the local optimum increases, as the matched ratio

\[
\sum_{i=1}^{300} \frac{N_i}{300}
\]
| #   | Matched (%) | Improvement (%) |
|-----|-------------|-----------------|
| 1   | 31.40       | 1.19            |
| 2   | 30.20       | 1.13            |
| 3   | 28.80       | 1.14            |

Table I. \( G^2(1; 30, 300, 6) \) simulation results.

| #   | Matched (%) | Improvement (%) |
|-----|-------------|-----------------|
| 1   | 25.60       | 1.34            |
| 2   | 28.60       | 1.43            |
| 3   | 28.80       | 1.44            |

Table II. \( G^2(2; 30, 300, 6) \) simulation results.

| #   | Matched (%) | Improvement (%) |
|-----|-------------|-----------------|
| 1   | 26.20       | 1.71            |
| 2   | 24.80       | 1.61            |
| 3   | 23.60       | 1.51            |

Table III. \( G^2(3; 30, 300, 6) \) simulation results.

Note that \( r = 1 \) has only \( M \) candidates in the search space, while they’re all covered by the default choice of \( f = M \), to match the computational cost of a conventional Greedy Algorithm.

Tables III IV and V present the extra value one can get with \( f = M, 2M, 3M \) for \( r = 3 \). While \( f = M \rightarrow 2M \) improves the probability to get of local optimum significantly by close to 10%, the value added on average is not that impressing. At the same time, \( f = 2M \rightarrow 3M \) shows less value, with a minor improvement of both metrics, which implies the selection of \( f \) is more of an art, as the trade-off between computational cost and added value.

3. \( G^2(\cdots ; \cdots ; M) \)

| #   | Matched (%) | Improvement (%) |
|-----|-------------|-----------------|
| 1   | 11.00       | 0.94            |
| 2   | 12.90       | 0.93            |
| 3   | 10.40       | 0.96            |

Table VI. \( G^2(1; 30, 300, 10) \) simulation results.

| #   | Matched (%) | Improvement (%) |
|-----|-------------|-----------------|
| 1   | 9.80        | 1.24            |
| 2   | 11.00       | 1.27            |
| 3   | 10.40       | 1.19            |

Table VII. \( G^2(3; 30, 300, 10) \) simulation results.

Comparing with \( M = 10 \) vs \( M = 6 \) data as shown in Sec. III B.1 with the same \( r = 1, 3 \), Table II vs Table VI and Table III vs Table VIII it shows that when \( M \) increases, the \( r = 1 \) option 3 times less likely to be trapped at local optimum, which also implies that Greedy Algorithm is very unlikely to be the global optimum \( P_{\text{opt Greedy}} < 10\% \).

Comparing with \( r = 1 \) vs \( r = 3 \) results for \( M = 10 \) in Tables VI vs VII again it shown that a larger \( r \) shows extra value to improve both the probability of getting out of the local optimum and the ability to find a better optimum based on the simple Greedy Algorithm.

4. \( G^2(\cdots ; W, N \cdots) \)

| #   | Matched (%) | Improvement (%) |
|-----|-------------|-----------------|
| 1   | 33.00       | 0.58            |
| 2   | 34.00       | 0.59            |
| 3   | 37.80       | 0.58            |

Table VIII. \( G^2(1; 100, 300, 6) \) simulation results.

Tables II vs VIII are different by 3 times on row size, i.e. the number of keywords space. Simulation suggests that both metrics get worse when \( K \) increases. \( P_{\text{trapped}} \) increases only a little bit, implying a Greedy Square Algorithm is slightly more likely to be trapped locally under a larger keywords space. Also the gain from the Greedy Square Algorithm is getting worse, reduced by a factor.
of 2. This is kind of expected, since each minimal iteration to add one more creative is exact in the keyword dimension, while the uncertainty lies mostly if not all in the creatives dimension. In this case, one might consider searching for a larger space, possibilities are to increase \( r, f \), i.e. the size of the search space.

| \# | Matched (%) | Improvement (%) |
|----|-------------|-----------------|
| 1  | 25.60       | 1.12            |
| 2  | 25.60       | 1.25            |
| 3  | 28.00       | 1.22            |

TABLE IX. \( G^2(1; 30, 1000, 6) \) simulation results.

Tables II vs IX are different by 3 times on column size, i.e. the number of creatives space. Simulation suggests that both metrics improve when \( N \) increases, an indication that a Greedy Square Algorithm is more likely to improve under a larger creatives space.

5. \( G^n(\cdots; \cdots) \)

Given the solution \( g^1 \) from conventional Greedy Algorithm, we should be able to run another round of Greedy Algorithm on top it, resulting in another solution \( g^2 \) from the Greedy Square Algorithm. Given the fact that \( g^2 \) is a solution resulting from the Greedy Algorithm, there’s essentially little or even no difference vs \( g^1 \). That’s, starting with the Greedy Algorithm solution in iteration \( i \), \( g^i \), one can always run another round of Greedy Algorithm to obtain a solution \( g^{i+1} \).

There might be two outcomes:

- \( g^{i+1} \) is worse than \( g^i \), keep \( g^i \) and stop here;

- \( g^{i+1} \) is equivalent to (starting in a new direction could find something new) or better than \( g^i \), keep \( g^{i+1} \);

Note that when the stopping rule of \( g^{i+1} \) being worse than \( g^i \) is triggered, \( G^n \) is equivalent to \( G^\infty \), which would be equivalent to the exact solution ideally, though it could be trapped somewhere in a local optimum in principle.

IV. CONCLUSIONS

We present a Generalized Greedy Algorithm, i.e. the Greedy-Power Algorithm \( G^n(r, f; W, N, M) \), characterized by three parameters: number of creatives to remove \( r \) from a given solution implying a difference of up to \( r \) creatives from original solution, number of branches to take for a given solution \( f \sim M \), and number of iterations for improvement \( n \) implying a Greedy Algorithm with a power of \( n \). With \( f \sim M \), we effectively impose that any following improvement step should have the same complexity of a conventional Greedy Algorithm, up to a factor of \( r \), and achieve solutions consistently better than the latter. As such, with twice (and an additional factor of \( r \) the cost of a Greedy Algorithm, one is likely to achieve decent added value from the Greedy-Power Algorithm, especially when the dimension \( M \) is large, when a Greedy Algorithm is more likely to be trapped at a local optimum, and corrected by the Greedy-Power Algorithm.

A. Future Directions

The setup of the Greedy-Power Algorithm \( G^n(r, f; W, N, M) \) is all done, while the effectiveness with real data is still under exploration, and it’d be interesting to see any significant difference of improvement from the random standard normal distribution. Also, it would provide more insights on the performance of Greedy Algorithm, and how it walks to the global optimum with a big \( n \), which would be equivalent to the \( n \to \infty \) when there is no more gain from the goal function.

At the same time, there could be other approaches for this problem setup, and one alternative is to find the exact solution is completely from a matrix perspective, which will be discussed in a separate paper in the near future.

V. REFERENCES

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition* (The MIT Press, 2009), 3rd ed., ISBN 0262033844, 9780262033848.