A Note on Bundle Profit Maximization

Guillermo Angeris
angeris@stanford.edu

Alex Evans
alex@placeholder.vc

Tarun Chitra
tarun@gauntlet.network

June 2021

Abstract

In the bundle allocation problem, a miner is given a fixed number of transactions which are to be included in a given block and a number of bundles which the miner can choose to include (or exclude) in this block. The miner earns profits from including each bundle in the block, but the bundles have a number of allocation constraints which must be accounted for. In this note, we give a simple formulation of the problem as an integer linear programming problem, and provide some basic extensions.

Introduction

First detailed in [DGK+19], Miner Extractable Value (MEV) is a term that refers to any excess profits that a miner can make based on transaction ordering. In decentralized systems such as blockchains, users submit a set of transactions and a fee to miners via a peer-to-peer gossip network. Miners collect these transactions and batch them into a totally ordered sequence which is then validated by a majority of miners and accepted as the next block. However in many chains, such as Ethereum, miners are allowed to choose both the set of transactions to be included and the sequence/ordering of transactions that are committed. If a miner is submitting an economically meaningful transaction, such as a trade, they can reorder the transactions to ensure their trade is executed first, which is also known as front-running. Since the original work describing MEV [DGK+19], there have been a number of novel and new forms of MEV involving flash loans [QZLG20], lending [KCCM20, QZG21], and sandwich attacks [ZQT+20]. MEV represents a form of value extraction that users cannot eliminate by simply modifying their transaction bidding behavior.

Fairness. Theoretically, MEV could lead to blockchain consensus instability and can force users to pay an additional tax on top of the expected transaction fee to have a transaction processed. This has led to a burst of research focused on guaranteeing 'fairness' in terms of transaction ordering and inclusion [Kur20, KZGJ20]. Fairness algorithms attempt to use cryptographic methods, such time-locked commitments to transaction ordering or pending transaction state, to enforce time-based 'fairness' guarantees.

MEV auctions. Alternatively, there is a line of work that MEV is endemic to blockchains and cannot be removed by purely cryptographic means. This line of work effectively purports that if instead of cryptographic removal of MEV, miners and users sharing of MEV profits would result is a stable equilibrium. In this world, pioneered by Flashbots, 'searchers' try to find optimal orderings for transactions and then bid for a 'bundle' of transactions to be executed by a miner in a particular order. This bid is mediated via an MEV auction—participants willing to pay miners for extra priority bid in an auction that is off chain. As such, MEV auctions are much more popular and have generated over $700 million dollars in excess revenue for miners in 2021 [MEV21].

Optimality. However, a natural theoretical question to ask is whether this auction is in some well-defined sense optimal. Currently, Flashbots auctions effectively execute transaction bundles by approximately solving a knapsack problem using a constraint solver. But is there any theoretically reason that we should expect an approximate integer linear program (ILP) solution to be 'optimal'? And how should optimality be depicted? Since MEV is defined in terms of the extractable value of all assets whose prices are affected by a reordering, any naïve notion of optimality depends on the maximum profit achievable by any set of transactions and bundles.

Summary. In this short note, we write out the first formal description of the optimal ILP for including transaction bundles within a single block. Our description focuses on MEV search action space consisting of frontrunning (placing a transaction right before a marked update), backrunning (placing a transaction right after a marked update), or sandwiching (placing a transaction before and after the update). We assume the precise gas simulation methods used in practice [The21] are performed as a preprocessing step, which decouples the allocation problem (the problem of finding the optimal bundle allocation) from the problem of correctly estimating the profits from individual bundles. Our formulation can be easily written in high level description languages for optimization such as CVXPY [AVDB18] and used in practice.

1 Definitions

In this section, we outline the basic definitions used throughout this note.

Transactions. A miner usually starts with a number of *transactions*, which we will write as some set T, that are to be included in the block. These transactions are provided by users of the chain, and can include Uniswap or Curve swaps, loans, oracle updates, among many other such transactions.

Bundles. A miner also receives a number of *bundles*, which are submitted by users. A bundle is an *action* (which we define later) with an associated transaction. Every bundle

also includes some *bid*; *i.e.*, how much the user is willing to pay in order for their bundle to be included in the block. The miner can decide which bundles and transactions are included in the block. The miner's profit from the bundles is then equal to the sum of the respective bids that were included in the block.

Actions. From before, every bundle associates an action with a transaction $t \in T$. The possible actions are: *frontrunning* the transaction t (executing a transaction right before t), *backrunning* the transaction (executing a transaction immediately after t), and *sandwiching* the transaction (executing both a transaction before and a transaction after t).

For a given transaction $t \in T$, it is only possible to either sandwich t or frontrun and backrun t. For example, if there are three bundles associated with transaction t, one of which backruns t, one of which frontruns t, and one which sandwiches t, the miner can choose to include the frontrunning and backrunning bundles, or the sandwiching bundle, but not both.

We call the space of these three actions A. We can now easily define a bundle as an action $a \in A$ associated with a transaction, $t \in T$ along with a bid amount $p > 0$; *i.e.*, the bundle is a triplet $(a, t, p) \in A \times T \times \mathbf{R}_+$. The set of all bundles will be given by $B \subseteq A \times T \times \mathbf{R}_+$.

Profit maximization. The remaining question is then: how can a miner choose which transactions are included in their block, in order to maximize their profits? In the following section, we will show that this problem can be phrased as a simple integer linear programming problem, which can often be solved in reasonable amounts of time on a modern computer.

2 Problem formulation

We formulate the problem of profit maximization as an integer linear program (ILP), which we will call the *bundle allocation* problem.

Set functions. We will write define the following functions for convenience. Here, $t \in T$ is a transaction and B is the set of all bundles.

We define $\mathbf{s}(t)$ as the set of bundles associated with transaction t that are sandwiches:

$$\mathbf{s}(t) = \{b \in B \mid b \text{ sandwiches transaction } t\},$$

and similarly for $\mathbf{f}(t)$, which are the frontrunning transactions associated with t, and $\mathbf{b}(t)$ which are the backrunning transactions associated with t. We will assume that B is indexed by $b = 1, 2, \ldots, n$, where n is the number of proposed bundles.

Problem statement. One simple way of writing the bundle allocation problem as an integer linear programming problem is as follows:

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & \displaystyle\sum_{b \in \mathbf{s}(t)} x_b + \frac{1}{2} \sum_{b \in \mathbf{f}(t) \cup \mathbf{b}(t)} x_b \leq 1, \quad t \in T \\ & \displaystyle\sum_{b \in \mathbf{f}(t)} x_b \leq 1, \quad \sum_{b \in \mathbf{b}(t)} x_b \leq 1, \quad t \in T \\ & x \in \{0,1\}^n. \end{array} \tag{1}$$

Here, $x \in \mathbf{R}^n$ is the optimization variable, where x_b is one if bundle b should be included in the current block, and 0 otherwise. The problem data is $c \in \mathbf{R}_+^{|B|}$, which is a vector such that $c_b \geq 0$ is the profit the miner receives for including bundle b in their block, and T is the set of transactions (not including bundles) which are to be included in this block.

Standard form. Problem (1) can be written slightly more compactly using matrix notation. To do this, we will define $m = |T|$, the total number of transactions, and the matrices $A, B, C \in \mathbf{R}^{m \times n}$ as:

$$A_{tb} = \begin{cases} 1 & b \in \mathbf{s}(t) \\ 0 & \text{otherwise,} \end{cases} \qquad B_{tb} = \begin{cases} 1 & b \in \mathbf{b}(t) \\ 0 & \text{otherwise,} \end{cases} \qquad C_{tb} = \begin{cases} 1 & b \in \mathbf{f}(t) \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

for each transaction $t \in T$ and bundle $b \in B$. Problem (1) can then be written in the following way, using these new definitions:

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & (A + (1/2)(B + C))x \leq \mathbf{1} \\ & Bx \leq \mathbf{1}, \quad Cx \leq \mathbf{1} \\ & x \in \{0,1\}^n, \end{array} \tag{3}$$

where $\mathbf{1}$ is the all-ones vector of the appropriate dimension, and $x \in \mathbf{R}^n$ is the optimization variable.

Interpretation. We can interpret the objective and constraints as follows. The objective $c^T x$ is simply the sum of the profits given by the bundles which have been included in the block. The first constraint implies that at most one sandwich bundle (associated with transaction t) is included in the block *or* at most two bundles that are either frontrunning or backrunning transaction t are included in the block. The second constraint implies that at most one frontrunning bundle is included and at most one backrunning bundle is included for each tranasction t. While the final constraint is simply that the entries of x are constrained to be Boolean.

Relaxations. In general, problem (1) is likely to be hard to solve other than for very small instances, due to the Boolean constraint on the entries of x. In many practical cases, though, relaxing the Boolean constraint to a box constraint, *i.e.*, $0 \leq x \leq \mathbf{1}$, can yield reasonable practical performance and reasonable solutions after some simple rounding schemes. In general, the optimal objective for this relaxed problem is always an upper bound on the best possible profit from the miner, while any rounding scheme gives a lower bound. This can be used to give a bound on how suboptimal a proposed bundle allocation is. For example, if the relaxation gives a profit of 1.2 ETH and a proposed allocation gives a profit of 1 ETH, then the suboptimality of the proposed allocation is, at most, $1.2/1 - 1 = 20\%$. In other words, it is possible to improve the proposed allocation by at most 20%.

2.1 Extensions.

There are a few simple, but very useful, extensions to problem (1).

Bundle constraints. For example, it is possible that a user might want to specify several bundles that must either all be included by the miner, all at once, or not at all. We can write this as subsets of bundles $B_i \subseteq B$, for $i = 1, \ldots, \ell$, where the entire subset of bundles B_i must be included by the miner, if any one bundle in B_i is included.

The new optimization problem is given by:

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & (A + (1/2)(B + C))x \leq \mathbf{1} \\ & Bx \leq \mathbf{1}, \quad Cx \leq \mathbf{1} \\ & Fx = Dy \\ & x \in \{0,1\}^n, \quad y \in \{0,1\}^\ell, \end{array} \tag{4}$$

where the optimization variables are $x \in \mathbf{R}^n$ and $y \in \mathbf{R}^\ell$, while the problem data are the matrices $A, B, C \in \mathbf{R}^{m \times n}$ as defined in (2) and the matrices $D \in \mathbf{R}^{\ell \times \ell}$ and $F \in \mathbf{R}^{\ell \times n}$:

$$D_{ij} = \begin{cases} |B_i| & i = j \\ 0 & \text{otherwise,} \end{cases} \qquad F_{i\ell} = \begin{cases} 1 & \ell \in B_i \\ 0 & \text{otherwise.} \end{cases}$$

In other words, D is a diagonal matrix whose diagonal entries are the size of the sets B_i, while F is a matrix such that $(Fx)_i$ gives the number of bundles in B_i that are to be included in the block. The constraint $Fx = Dy$ simply states that either all $|B_i|$ bundles are to be included or exactly 0 are to be included, for each possible i.

Gas constraints. Another possible (and very simple) extension is the inclusion of a total gas constraint on the optimization problem. For example, each bundle $b \in B$ may use some maximum amount of gas when included in the block, given by $g_b \geq 0$. We can easily append the constraint that the total maximum amount of gas used by the bundles is no more than

the amount of gas remaining after the transactions, but not the bundles, are executed; *i.e.*, that $g^T x \leq M$, where $M \geq 0$ is the remaining amount of gas. We note this could be a difficult quantity to get a reasonable bound on, as the gas used by a transaction could change drastically when bundles are included in the block. There are other possible ways of applying more careful accounting methods, but we do not discuss them here.

3 Conclusion

In this note, we've provided a simple, but very general formulation of the bundle allocation problem that a miner might wish to execute in order to maximize profit. While the problem is likely NP-hard in general, we suspect that most integer linear programming solvers (and even linear programming relaxations) might have very good performance on practical instances.

References

[AVDB18] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[DGK+19] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges. *arXiv:1904.05234 [cs]*, April 2019.

[KCCM20] Hsien-Tang Kao, Tarun Chitra, Rei Chiang, and John Morrow. An analysis of the market risk to participants in the compound protocol. In *Third International Symposium on Foundations and Applications of Blockchains*, 2020.

[Kur20] Klaus Kursawe. Wendy, the good little fairness widget: Achieving order fairness for blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 25–36, 2020.

[KZGJ20] Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. Order-fairness for byzantine consensus. In *Annual International Cryptology Conference*, pages 451–480. Springer, 2020.

[MEV21] Mev21 explore v0, 2021. `https://explore.flashbots.net`.

[QZG21] Kaihua Qin, Liyi Zhou, and Arthur Gervais. Quantifying blockchain extractable value: How dark is the forest? *arXiv preprint arXiv:2101.05511*, 2021.

[QZLG20] Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais. Attacking the defi ecosystem with flash loans for fun and profit. *arXiv preprint arXiv:2003.03810*, 2020.

[The21] TheGoStep. Fr: Bundle merging. `https://github.com/flashbots/pm/discussions/27`, 2021.

[ZQT+20] Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc V Le, and Arthur Gervais. High-frequency trading on decentralized on-chain exchanges. *arXiv preprint arXiv:2009.14021*, 2020.