# Hardness of the Pricing Problem for Chains in Barter Exchanges

Benjamin Plaut[a], John P. Dickerson[a], Tuomas Sandholm[a]

[a]*Computer Science Department*
*Carnegie Mellon University*

## Abstract

Kidney exchange is a barter market where patients trade willing but medically incompatible donors. These trades occur via cycles, where each patient-donor pair both gives and receives a kidney, and via chains, which begin with an altruistic donor who does not require a kidney in return. For logistical reasons, the maximum length of a cycle is typically limited to a small constant, while chains can be much longer. Given a compatibility graph of patient-donor pairs, altruists, and feasible potential transplants between them, finding even a maximum-cardinality set of vertex-disjoint cycles and chains is NP-hard. There has been much work on developing provably optimal solvers that are efficient in practice. One of the leading techniques has been *branch and price*, where column generation is used to incrementally bring cycles and chains into the optimization model on an as-needed basis. In particular, only positive-price columns need to be brought into the model. We prove that finding a positive-price chain is NP-complete. This shows incorrectness of two leading branch-and-price solvers that suggested polynomial-time chain pricing algorithms.

*Keywords:* kidney exchange, matching markets, integer programming, branch and price

## 1. Introduction

In barter markets, participants swap items for items rather than for money. Barter markets have been organized for many categories of items such as holiday home time, nurse shifts, used goods, and even shoes. Kidney exchange is an organized barter market where patients in need of a kidney trade their paired but incompatible donors with other participants in the exchange [1]. Mathematically, a kidney exchange is typically modeled as a directed *compatibility graph* $G = (V, E)$. The set of vertices $V$ is partitioned into $P$ and $A$, where vertices in $P$ represent patient-donor pairs and vertices in $A$ represent altruistic donors, who enter the exchange without a paired patient. For each $u, v \in P$, the edge $(u, v)$ exists if the donor of pair $u$ is compatible with the patient of pair $v$. Similarly, for each $a \in A$ and $v \in P$, the edge $(a, v)$ exists if altruist $a$ is compatible with the patient of pair $v$. These edges may also have weights, representing the relative value of a potential transplant.

Given a compatibility graph $G$, the *weighted clearing problem* is to find a maximum-weight vertex-disjoint set of cycles of length at most $L$ and chains of length at most $K$. Even the maximum-cardinality clearing problem with $L \geq 3$ is NP-hard [2]. In practice, $L$ is limited to a small constant (typically $L = 3$) due to the logistical difficulty of arranging all transplants in a cycle simultaneously; however, due to the non-simultaneous execution of chains, $K$ is often much greater than $L$—and can even be allowed to grow with $|V|$.

Integer programming techniques power optimal clearing engines in fielded kidney exchanges. A leading approach to solving these integer program models—which can be quite large due to the $O(|P|^L)$ cycles and $O(|A||P|^K)$ chains in a compatibility graph—is *branch and price* [2]. Branch and price [3] is a technique where only a subset of the columns (cycles and chains) are kept in the model, and promising columns are incrementally generated and added to the reduced model until optimality can be proven. Such promising columns are found by solving the *pricing problem*, which searches for at least one positive price column (i.e., variable) to add to the model—or shows that none exist. Once no more positive price variables exist, optimality has been proven for that node in the branch-and-bound search tree, and the search can proceed further in the tree.

Quickly solving the pricing problem at nodes in the search tree is important for overall runtime. Recently, it was shown that determining whether a positive price *cycle* exists can be solved in polynomial time [4, 5]. Both Glorie *et al.* [4] and Plaut *et al.* [5] also use a variant of their cycle-pricing algorithms for chains. In this paper, we show that not only are those latter algorithms incorrect, but the underlying problem—finding a positive price *chain*—is, in fact, NP-complete.

## 2. The pricing problem in kidney exchange

We now formally define the pricing problem in the context of kidney exchange, but it applies to other barter markets as well. The pricing problem is to find at least one positive price cycle or chain, or show that none exist. The price of a cycle or chain $c$ is $\sum_{(u,v)\in c} w_{(u,v)} - \sum_{v\in c} \delta_v$, where $w_{(u,v)}$ is the weight of edge $(u,v)$, and $\delta_v$ is the dual value of vertex $v$ in the linear program relaxation. Glorie *et al.* [4] and Plaut *et al.* [5] show how determining whether a positive price cycle in the compatibility graph $G = (V, E)$ exists is equivalent to finding a negative weight cycle in a *reduced graph* $G' = (V, E')$, where each edge $e' = (u, v) \in E'$ exists if and only if $(u, v) \in E$, and $e'$ has *reduced weight* $r_{(u,v)} = \delta_v - w_{(u,v)}$.

A similar equivalence holds for chains. We must be careful, however, since the number of vertices in a chain exceeds the number of edges by 1. We now define $r_{(u,v)}$ as follows:

$$
r_{(u,v)} = \begin{cases} \delta_v - w_{(u,v)} & u \in P \\ \delta_u + \delta_v - w_{(u,v)} & u \in A \end{cases}
$$

Since an outgoing edge from an altruist will only ever be used in a chain, this ensures that a chain has positive price in $G$ if and only if it has negative weight in the reduced graph $G'$.

## 3. Counterexample to two prior algorithms

In this section, we provide counterexamples to the pricing algorithms of both Glorie *et al.* [4] and Plaut *et al.* [5]. Both previous algorithms use Bellman-Ford-style search in the reduced graph, initiated from each altruist as the source, to find negative-weight chains. Ideally, we would like to find the *shortest* paths using each vertex at most once, but this is NP-hard in the presence of negative cycles [5]. However, we need not find the shortest paths

beginning at each altruist: we only need to determine whether there exists any negative path starting at any altruist.

In the presence of negative cycles, traditional Bellman-Ford may generate paths with internal loops, which are invalid in our context. Plaut *et al.* [5] handle this by preventing Bellman-Ford from looping during execution. As a result, the generated paths may not be the shortest, and a given negative chain may not be found.

For the version of the algorithm for cycles, Plaut *et al.* [5] show that although there may be negative cycles that are not found, at least one negative cycle will be found, if any exist. Their proof of the version of the algorithm for chains is incorrect in general, however, as it implicitly assumes that the chain length cap and cycle length cap are equal.

Plaut *et al.* [5] gave a counterexample to the algorithm of Glorie *et al.* [4]. Figure 1 gives a counterexample to the algorithm of Plaut *et al.* [5]; this is also a counterexample to the original algorithm due to Glorie *et al.* [4].
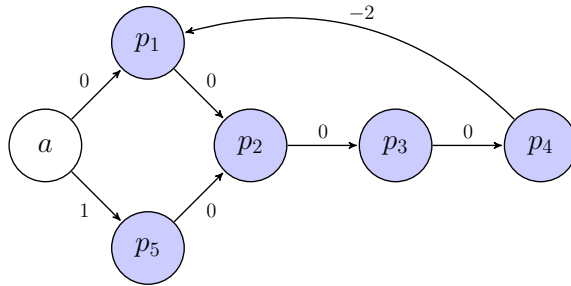


Figure 1: Example where the algorithm of Plaut *et al.* [5] fails to find a negative chain for $L = 3$ and $K = 5$, although one exists. Here, vertex $a$ is an altruist ($a \in A$) and the rest of the vertices are incompatible donor-patient pairs ($\{p_1, \ldots, p_5\} \in P$).

For $L = 3$ and $K = 5$, there are no valid negative cycles in the reduced graph, and there is a single valid negative chain in the graph: $(a, p_5, p_2, p_3, p_4, p_1)$. Although $(p_1, p_2, p_3, p_4)$ is a cycle with negative weight, it exceeds the cycle length cap of $L = 3$, and thus is invalid.

In the second iteration of the algorithm due to Plaut *et al.* [5], vertex $p_2$ would store as its most promising predecessor the path $(a, p_1, p_2)$ with weight $w\left[(a, p_1, p_2)\right] = 0$, instead of $(a, p_5, p_2)$ with less promising weight $w\left[(a, p_5, p_2)\right] = 1$. However, this causes the algorithm to miss the overall negative chain that would be found otherwise at iteration 5, since it cannot reuse vertex $p_1$ (and thus cannot use the sole negative-weight edge with sink $p_1$). Critically, even though the path $(a, p_5, p_2)$ was not promising at an earlier iteration, following it instead of the more immediately promising $(a, p_1, p_2)$ would have led to a negative-weight chain—in this case, the only negative-weight chain. The initial algorithm due to Glorie *et al.* [4] would also incorrectly return that no negative-weight chains exist, by similar reasoning.

This shows a correctness error in both Glorie *et al.* [4] and Plaut *et al.* [5]. In the next section, we show that in general such polynomial-time approaches are hopeless: determining whether a positive price chain exists is NP-complete.

## 4. Main result

We define the *negative chain problem* as follows: given a directed graph $G = (V, E)$, where $V = P \cup A$, is there a path (using each vertex at most once) of negative weight, using

at most $K$ edges, and starting at some vertex $a \in A$? We call such a path a negative chain.

**Theorem 1.** *Deciding whether a negative chain exists is NP-complete.*

*Proof.* The negative chain problem is trivially in NP: simply sum the edge weights in a proposed path and check its sign. To show NP-hardness, we reduce from the directed Hamiltonian path problem. Given some graph $H = (V, E)$, the directed Hamiltonian path problem asks whether there exists a directed path that visits each vertex exactly once. Let $n = |V|$. Construct the graph $G$ as follows: set $w_e = -1$ for each $e \in E$, and add a vertex $a$ with an edge $(a, v)$ with $w_{(a,v)} = n - 2$ for each $v \in V$. Figure 2 gives an example of the construction of the graph $G$ for the proof. Let $P = V$, $A = \{a\}$, and $K = n$.
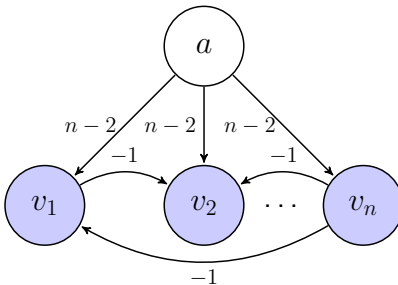
Figure 2: Example construction for the proof of Theorem 1.

Suppose $h$ is a Hamiltonian path in $H$ starting at $v_i$. Let $c = (a, v_i) \cup h$. Since $h$ has exactly $n - 1$ edges, $c$ contains $n$ edges, thereby satisfying the length constraint. Since $h$ visits each $v_i$ exactly once and never visits $a$, $c$ visits each vertex in $G$ at most once. Finally, since $h$ has weight $1 - n$, $c$ has weight $n - 2 + 1 - n = -1$. Therefore $c$ is a negative chain in $G$.

Suppose $c$ is a negative chain in $G$. Then $c$ must begin at $a$, so we can write $c = (a, v_i) \cup h$, for some $v_i \in V$ and path $h$. Let $m$ be the number of edges in $h$. Then $w_c = n - 2 - m$. Since $w_c < 0$, we have $m > n - 2$. Since $c$ can use each vertex at most once, we have $m \leq n - 1$. Therefore $m = n - 1$. Since $h$ has $n - 1$ edges, $h$ visits every vertex in $V$ exactly once, making it a valid Hamiltonian path in $H$. $\square$

The general pricing problem (where both cycles and chains are included) is to find a positive price (negative weight) cycle of length at most $L$ or a positive price (negative weight) chain of length at most $K$, or show that none exist. Note that solving the general pricing problem does not necessarily solve the negative chain problem. If $X$ is the set of negative chains and $Y$ is the set of negative cycles, the general pricing problem is to determine whether $X \cup Y = \emptyset$. The negative chain problem is to determine whether $X = \emptyset$: however, determining whether $X \cup Y = \emptyset$ does not necessarily determine whether $X = \emptyset$.

To show that the general pricing problem is NP-hard, we modify the above construction by expanding each edge in $H$ to a series of $L$ edges whose weights sum to $-1$. Then any cycle in $G$ has length at least $2L$, which violates the length constraint. Since there are no valid negative cycles in $G$, the pricing problem becomes equivalent to the negative chain problem. Therefore, the general pricing problem is also NP-hard. Since the general pricing problem is also trivially in NP, it is NP-complete.

## 5. Conclusion & implications

We discussed branch-and-price-based approaches to the kidney exchange problem, and showed that solving the pricing problem for chains, and thereby the pricing problem for cycles and chains jointly, is NP-complete. This shows a correctness error in two leading branch-and-price-based solvers. The results apply to other barter exchanges as well, as long as they use chains (potentially with cycles as well).

Our hardness results show that a different approach for handling chains is necessary. Dickerson *et al.* [6] introduce models where chains are represented by position-indexed edge variables. Since there are only a polynomial number of edge variables, they can be fully enumerated, removing the need for branch and price for chains. Cycles in these models can still be handled via branch and price, or via a different scheme.

## Acknowledgements

## References

[1] A. Roth, T. Sönmez, U. Ünver, Kidney exchange, Quarterly Journal of Economics 119 (2) (2004) 457–488.

[2] D. Abraham, A. Blum, T. Sandholm, Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges, in: Proceedings of the ACM Conference on Electronic Commerce (EC), 2007, pp. 295–304.

[3] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, P. H. Vance, Branch-and-price: Column generation for solving huge integer programs, Operations Research 46 (3) (1998) 316–329.

[4] K. M. Glorie, J. J. van de Klundert, A. P. M. Wagelmans, Kidney exchange with long chains: An efficient pricing algorithm for clearing barter exchanges with branch-and-price, Manufacturing & Service Operations Management (MSOM) 16 (4) (2014) 498–512.

[5] B. Plaut, J. P. Dickerson, T. Sandholm, Fast optimal clearing of capped-chain barter exchanges, in: AAAI Conference on Artificial Intelligence (AAAI), 2016.

[6] J. P. Dickerson, D. Manlove, B. Plaut, T. Sandholm, J. Trimble, Position-indexed formulations for kidney exchange, in: Proceedings of the ACM Conference on Economics and Computation (EC), 2016.