An improved algorithm for multi-way trading for exchange and barter

Randy M. Kaplan

Computer Science, Kutztown University of Pennsylvania, Kutztown, PA 19530, United States

ABSTRACT

Most e-commerce exchange systems today facilitate a single simple transaction. Systems allow the trade of an object or resource for exactly one other object or resource. The trade could also be for some quantity of one object or resource for another, but usually all of the objects traded are of a kind, for example, trading 10 iPods. The common type of exchange involves money for an object. There are other kinds of exchanges though. These include barter exchanges in which an object is traded for another object. In this scenario there is a space of available objects and a space of desired objects. When an available object matches a desired object a trade can be made. Usually this type of trade is bi-directional. One trader has an item that a second trader wants and the second trader has the item that the first trader wants. This trade is a simple two-party trade. Another kind of trade involves multi-way trade in which multiple parties are involved, along with an intermediary. Multi-way trading is the process of constructing a trade that involves two original or initial traders but also has intermediary traders to facilitate the original trade. This research presents an algorithm for supporting multi-way trades. It describes a graph data structure and then searches the graph for paths to effect the trade. The algorithm also identifies a sequence of trades through at least one intermediate trader that will complete the trades of the two original traders.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

A class of mathematical problems called *matching problems* (Kleitman 2008) has been of interest in mathematics, operations research, and optimization for some time. In any matching problem there is a group of *objects*. The idea is to connect the objects in a pair-wise fashion with one another. Consider couples that come to a dance. Imagine that each of the couples is married, and a person distributes cards that specify who will dance with whom. If two of these dancing assignments result in a couple who is also married, we call this a *match*.

The quest in mathematics has been to find an algorithm given a set of objects, that will efficiently compute the maximum number of pairings or matches that can be produced from the set of objects. Vertices in a graph represent the set of objects. The pairings are represented by edges connecting two objects. No edge can coincide with more than two vertices, one at each end of the edge. The time required to compute the set of possible matching grows exponentially with the number of vertices in the graph. An algorithm that computed matches in a brute-force manner would not be practical. Edmonds (1965) proposed an approach to computing matches in polynomial time. The method involves constructing a path between two vertices in the graph using an alternating sequence of matched edges and non-matched edges. (See Fig. 1.)

A couple of important comments are in order about this algorithm. First, it is an *all-pairs algorithm*. It does not distinguish between what we would call correct matches, as in the dance example given previously, and incorrect matches: it simply computes matches. Second, the procedure works fine for pairings but becomes computationally non-polynomial when we begin to consider matchings that involve more than two vertices. For now, we observe that the matching problem is related to the multi-way trading problem described here.

2. Multi-way trading: an example

Multi-way trading is the process of trading one object for another object through a series of trades that eventually results in the original traders obtaining what they desire. Each trade that is part of a multi-way trade is called a *step*. A *simple trade* is a multi-way trade when the number of steps is one. (See Fig. 2.)

For this trade, Trader A and Trader B can consummate the trade. Consummation of the trade means that Trader A gives his iPod to Trader B, and in turn Trader B gives her camera to Trader A. Now each of the traders has their desired objects. Once this state is obtained, the trade is consummated.

A multi-way trade involves $n + 2$ traders, and $n > 0$ is the total number of steps of the trade. Here, n represents the number of intermediate traders and $n + 2$ is the total number of traders in the trade. In a multi-way trade, the n intermediate traders need

E-mail address: kaplan@kutztown.edu

Maximum-Matching (G)

     $M \leftarrow \{\}$
     repeat
         if there is an augmenting path with respect to M
         then
             Let P be an augmenting path with respect to M
             $M \leftarrow M \oplus P$
         else
             return M and halt.

Fig. 1. Edmond's matching algorithm. *Note*: Edmonds's result showed that the matching can be accomplished in $O(VE)$ time, where V is the number of vertices of the graph and E is the number of edges (Karp 2006). The general matching problem has garnered significant attention and more efficient algorithms have been formulated. For example, Gabow and Tarjan (1991) identify an improved algorithm with a time bound of $O(\sqrt{V}E)$.
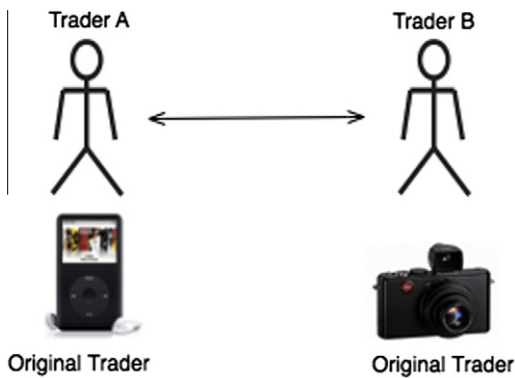
Fig. 2. A simple trade. Trader A is offering an iPod and desires a camera. Trader B is offering a camera and desires an iPod.

not be distinct. A single trader may supply more than one object to be traded. Let us assume that Trader A offers an iPod and desires a camera. Trader B offers a camera and desires clothing. Is there a way for Trader A to complete his desired trade?

The set of all possible trades is called the *trading pool.* Imagine that in the trading pool there is a Trader C that is offering clothing and desires shoes. Another Trader D is offering shoes and desires an iPod. The trade sequence to achieve the original trade is as follows. (See Fig. 3.)

This trade sequence represents a multi-way-trade consisting of three trading steps. At each step, Trader A acquires an object and eventually obtains the desired object, the camera. Although this sequence depicts Trader A as the recipient of each traded object, there is no reason to do this in practice.[1]

Recall that multi-way trading is a variant of the matching problem. An important difference between the multi-way trading problem and the general matching problem is that we know *a priori* the precise constraints on a particular trade. As such, we are not interested in computing all matches but rather a subset of the possible matches: just those matches that satisfy the trader's requirements. The algorithm presented here uses this information to construct a representation of the possible trades in the form of a tree called a *trade sequence tree.* The problem reduces to searching the tree for a path that begins with the object to trade and ends with the desired object. It is well established that the *n*-way matching problem is NP-complete (Karp 1972). A significant advantage of the algorithm presented here is that regardless of *n*, the algorithm performs in

polynomial time, $O(|T|^l)$, where T is the set of possible trades and l is the maximum length of the trade. A second significant advantage of the algorithm presented here is that the matching does not have to be recomputed each and every time a trade is computed. The tree can be modified to reflect the current state of the trading pool.

3. Description of the problem

If the only trades in the trading pool were those that were needed to complete the trade, then the problem would be trivial, as we discussed earlier. Unfortunately this will not be the case. The trading pool will contain an arbitrary number of trades. In order to locate the trading sequence that would satisfy the desired trade – if one exists – we could systematically create trading sequences of all lengths. We would start by creating all trading sequences of length 2. If no trade sequence of length 2 satisfies the desired trade, then all trading sequences of length 3 will be created until a sequence fulfills the desired trade. This continues through successively larger sequence lengths until a trading sequence is found or until a limit on the trade sequence length is reached. In the worst case the longest trade sequence that could be created is the same as the number of trades in the trading pool. Therefore the worst case running time of this process is $O(n^n)$ which is clearly unacceptable, and could not be used in any practical sense. The problem is how do we search the n^n possible paths?

The *set of possible trades P* consists of tuples *p*, which we call a *trading pair.* In the following discussion, *O* denotes an object in the trading pool, *D* denotes objects in a trader's desired trade, and *T* denotes the trading pool. The subscript *d* is used to mark an object as a desired object in the trading pool. The subscript *o* is used to mark an object in the trading pool as one that is offered or available for trade. Thus O_d denotes a desired object and O_o denotes an object available for trade. A trading pair tuple consists of a desired object O_d, and an offered object O_o, as in $\langle O_d, O_o \rangle$. The trader, who is the person wanting to make the trade, has a desired trade that he wants to have consummated. The desired trade is also a tuple consisting of $\langle D_d, D_o \rangle$.

Suppose we have set of tuples, *T*, representing all of the possible trades that can take place the trading pool:

$$T = \{\langle O_{d1}, O_{o1} \rangle, \langle O_{d2}, O_{o2} \rangle, \ldots, \langle O_{dn}, O_{on} \rangle\}$$

In the trivial case there exists some $\langle O_{di}, O_{oi} \rangle$ such that $\langle O_{di}, O_{oi} \rangle \in T$ and $O_{oi} = D_d$ and $O_{di} = D_o$. In the non-trivial case there is no tuple $\langle O_{di}, O_{oi} \rangle$ that satisfies these criteria, so we need to look for a trading sequence in the multi-way trade that will achieve the same result. If *S* is an ordered set of trading pairs that consummates the trade, we are searching for the set *S*, as follows:

$$S = \{\langle O_{d1}, O_{o2} \rangle, \langle O_{d2}, O_{o3} \rangle, \langle O_{d3}, O_{o4} \rangle, \ldots, \langle O_{dn-1}, O_{o1} \rangle\}$$

In the above sequence $\langle O_{d1}, O_{o2} \rangle$ partially satisfies $\langle O_{d2}, O_{o3} \rangle$ because the object offered in the first pair O_{o2} is the desired object of the second pair O_{d2}. Trade 2 is partially satisfied by Trade 1. At the end of the sequence the previous trade satisfies the final trade, and the offered object O_{o1} of this trade satisfies the originally desired object O_{d1}. The multi-way trading algorithm identifies the set *S* of trade tuples that satisfies the originally desired trade.

4. The multi-way trading algorithm

The multi-way trading problem is one of finding the set *S* of trading pairs, $S = \{\langle O_{d1}, O_{o2} \rangle, \langle O_{d2}, O_{o3} \rangle, \langle O_{d3}, O_{o4} \rangle, \ldots, \langle O_{dn-1}, O_{o1} \rangle\}$. *S* represents a sequence of trades that will ultimately fulfill a trader's originally specified desired trade, $\langle O_d, O_o \rangle$. We need to extract this sequence from a set of arbitrarily many trading pairs. *P* is the set of all of the trading pairs. To create a tree *T* whose root is O_o, the object offered by the trader desiring the object O_d, we do the following:

---

[1] In practice, each trader will be given appropriate instructions. Consider the trade in Fig. 3. Here are the instructions that would be given to the traders. Trader A: Give your iPod to Trader D. Trader D: Give your shoes to Trader C. Trader C: Give your clothes to Trader B. Trader B: Give your camera to Trader A. At the end of this sequence, each trader will ends up with their desired object.