BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **20**, No 5
Special issue on Innovations in Intelligent Systems and Applications
Sofia • 2020 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2020-0042

Optimal Semi-Competitive Intermediation Networks

Amelia Bădică[1], *Costin Bădică*[1], *Maria Ganzha*[2], *Mirjana Ivanović*[3], *Marcin Paprzycki*[4]

[1]*University of Craiova, Craiova, Romania*
[2]*Warsaw University of Technology, Warsaw, Poland*
[3]*University of Novi Sad, Novi Sad, Serbia*
[4]*Polish Academy of Sciences, Warsaw, Poland*
E-mails: ameliabd@yahoo.com cbadica@software.ucv.ro Maria.Ganzha@ibspan.waw.pl mira@dmi.uns.ac.rs marcin.paprzycki@ibspan.waw.pl

***Abstract*:** *In this work we address the problem of optimizing collective profitability in semi-competitive intermediation networks defined as augmented directed acyclic graphs. Network participants are modeled as autonomous agents endowed with private utility functions. We introduce a mathematical optimization model for defining pricing strategies of network participants. We employ welfare economics aiming to maximize the Nash social welfare of the intermediation network. The paper contains mathematical results that theoretically prove the existence of such optimal strategies. We also discuss computational results that we obtained using a nonlinear convex numerical optimization package.*

***Keywords*:** *Social welfare, Intermediation, Profitability, Nonlinear convex optimization, Directed acyclic graph.*

1. Introduction

Current developments in Computing and Information Technology support the practical application of graph and network models in many areas of science and engineering, including the emergent fields of e-Commerce and e-Business. Researchers in Economics and Computer Science are interested to understand how intermediation networks conceptually emerge and technically develop. Economists are focused on intermediation in decentralized trading markets comprising autonomous business agents. Meeting, interaction and trading of buyers and seller is now facilitated either through direct contact, or via intermediation agents, with reduced cost and increased gain. Computer Scientists have developed digital decentralized markets where business partners assisted by software agents can register their capabilities, search for potential partners, and involve in trading activities [6, 13].

Current practices in the management of supply and distribution activities assume the engagement of business agents in inter-related and concurrently operating supply and distribution channels. In our recent research work we are interested in the modelling and analysis of such complex business processes as semi-competitive intermediation networks. Our results on this subject are presented in papers [4, 5] and synthesis paper [12] and refer to the following aspects of the problem: i) the introduction of a domain-independent process model, grounded on Directed Acyclic Graphs (DAG hereafter), and ii) the development of necessary and sufficient conditions that guarantee profitable pricing strategies of involved actors in the intermediation process.

Observe that most of existing research in intermediation processes is developed in a competitive environment. However, we have noticed that actually intermediation networks are serving supply and distribution chains operating in semi-competitive, rather than competitive environment. This means that while participating agents are collaborating so that the underlying business process is supporting the service of its external customers, the agents are also self-interested with the goal to maximize both their immediate private revenue, as well as their longer-term welfare [12].

Our proposed model is relevant for both single and multiple company settings. It covers also B2B models of industrial consortia and private industrial networks that perform network-based B2B e-Commerce in privately, as well as collectively own "extended enterprises". Such complex organizations include carefully selected multiple independent business participants committed to tightly coordinate efficiently and collaborate for supporting industry-wide goals, quite often serving a single and very large manufacturer [14].

All participants in the intermediation business process define an "agent society". Social choice theory postulates that social choice is governed by a social welfare function mapping tuples of "individually preferred" outcomes of the participants to the "socially preferred" outcome. Four rationality axioms that must be satisfied by an acceptable social welfare function are proposed based on common-sense principles. This enabled the proof that there is a unique mathematical function that satisfies these axioms, known as the Nash social welfare function [9].

Let us assume that participants' choices are determined by transaction prices such that their social welfare can be quantitatively evaluated as Nash social welfare index. Our most important result is the introduction of optimal pricing strategies of transaction participants as nonlinear convex optimization problem [7]. We obtain theoretical results regarding the existence of such optimal pricing strategies in semi-competitive intermediation networks. Our results are also supported by experimental evidence.

2. Intermediation networks

We start by reviewing our definition of collectively profitable intermediation processes [12]. Central to this definition is the concept of *intermediation DAG*. We consider a simplified form of this definition, by discarding DAG annotations of exchanged products, as they are not relevant for the results reported in this paper.

Definition 1. Intermediation DAG. Let us consider three finite, nonempty and pairwise disjoint sets – $\mathcal{S}$, $\mathcal{B}$, and $\mathcal{I}$ be of sellers, buyers and intermediaries. An *intermediation DAG* is represented by the triple $\mathcal{G} = \langle \mathcal{V}, \mathcal{A}, p \rangle$ such that:

1. The set of vertices is defined by $V = \mathcal{S} \cup \mathcal{B} \cup \mathcal{I}$.
2. The sets of incoming arcs to $\mathcal{S}$ and outgoing arcs from $\mathcal{B}$ are empty.
3. $p : \mathcal{S} \cup \mathcal{B} \cup \mathcal{A} \to [0, +\infty)$ is a function that annotates the elements of G with pricing information, as follows.

a. If $s \in \mathcal{S}$ then $p(s) = \sigma_s > 0$. Here $\boldsymbol{\sigma}$ denotes the vector of *seller limit prices*.

b. If $b \in \mathcal{B}$ then $p(b) = \beta_b > 0$. Here $\boldsymbol{\beta}$ denotes the vector of *buyer limit prices*.

c. If $t = (u, v) \in \mathcal{A}$ denotes a transaction then $p(t) = p_{u,v} > 0$ is the transaction price for which seller agent u agrees to sell its products to buyer agent v.

Limit prices are explained in what follows. Each participant agent (with seller, buyer or intermediation role) that is part of an intermediation transaction is selling and/or buying a given set of products. Their strategies are governed by suitable chosen and pre-defined limit prices. Let us denote with σ_s the limit price of seller s. So agent s agrees to sell its products only at price $p \geq \sigma_S$. Similarly, let us denote with β_b the limit price of buyer b. So agent b agrees to purchase its set of products only at price $p \leq \beta_b$.

Let us consider the example intermediation DAG shown in Fig. 1. Observe that: $\mathcal{S} = \{1, 2\}$, $\mathcal{B} = \{5, 7, 8\}$, $p(1) = \sigma_1$, $p(5) = \beta_5$, and $p((3, 6)) = p_{3,6}$.

Let us define functions "in" and "out":

$$\text{in} : \mathcal{V} \to 2^{\mathcal{V}},\ \text{in}(v) = \{u \mid (u, v) \in \mathcal{A}\},$$
$$\text{out} : \mathcal{V} \to 2^{\mathcal{V}}, \text{out}(v) = \{u \mid (v, u) \in \mathcal{A}\}. \quad (1)$$

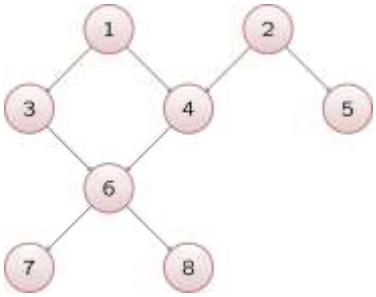

Fig. 1. Sample DAG-based intermediation process

We can now define the utility $u_v = u(v)$ of each agent node $v \in \mathcal{V}$:

the utility of seller agent $s \in \mathcal{S}$ is

$$u(s) = -\sigma_s + \sum_{v \in \text{out}(s)} p_{s,v}; \quad (2)$$

the utility of buyer agent $b \in \mathcal{B}$ is

$$u(b) = \beta_b - \sum_{v \in \text{in}(b)} p_{v,b}\ ; \quad (3)$$

the utility of intermediation agent $i \in \mathcal{I}$ is

$$u(i) = \sum_{v \in \text{out}(i)} p_{i,v} - \sum_{v \in \text{in}(i)} p_{v,i}. \quad (4)$$

Participant agent $v \in \mathcal{V}$ is called *profitable* if and only if the outcome of its transaction is a positive utility, i.e., $u(v) \geq 0$.

Definition 2. Collective profitability. An intermediation DAG $\mathcal{G}$ is called *collectively profitable* if and only if there exist positive transaction prices p that annotate $\mathcal{G}$ in such a way that for each agent v captured by a vertex of $\mathcal{G}$ we have $u(v) \geq 0$.

Our preliminary results include a set of necessary and sufficient collective profitability conditions that must be satisfied by the vectors of limit prices $\boldsymbol{\sigma}$ and $\boldsymbol{\beta}$ [12]. Here, we assume that these conditions hold, i.e., our DAG is collectively profitable, and we focus on studying the maximization of social welfare as mathematical optimization problem.

3. Optimal social welfare

Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{A}, p \rangle$ be a collectively profitable intermediation DAG. This means that $u_v = u(v) \geq 0$ for each participant agent $v \in \mathcal{V}$. Let $n \geq 1$ be the total number of participants (i.e., the cardinal of finite set $\mathcal{S}$) and $e \geq 1$ be the total number of transactions (i.e., the cardinal of finite set $\mathcal{A}$). Hence, we can number participants with $1, 2, \ldots, n$ and transactions (in lexicographical order) with $1, 2, \ldots, e$.

The Nash social welfare function for the society of participant agents defined by intermediation graph $\mathcal{G}$ by:

$$U(\mathbf{p}) = \sum_{i=1}^{n} \log u_i. \tag{5}$$

Observe that if graph $\mathcal{G}$ is collectively profitable then U is well defined by (5), as logarithm is defined for positive real numbers and utilities u_i are positive for all $i = 1, 2, \ldots, n$ (assuming that $\log 0 = -\infty$). Let $\mathbf{b} \in \mathbb{R}^{n \times 1}$ be the vector of signed limit prices, defined by:

$$b_i = \begin{cases} -\sigma_i & i \in \mathcal{S}, \\ \beta_i & i \in \mathcal{B}, \\ 0 & i \in \mathcal{I}. \end{cases} \tag{6}$$

We denote with $D \in \mathbb{R}^{n \times e}$ the incidence matrix [1] of G, defined by:

$$D_{i,j} = \begin{cases} 1 & \text{if the head of arc } j \text{ is node } i\,, \\ -1 & \text{if the tail of arc } j \text{ is node } i, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Vector $\mathbf{u}$ representing the utilities of participant agents can be defined by:

$$\mathbf{u} = D \cdot \mathbf{p} + \mathbf{b}. \tag{8}$$

Collective profitability condition $\mathbf{u} \geqslant 0$ can be combined with positivity condition of transaction prices $\mathbf{p} \geqslant 0$ thus producing the next equation. Note that the next equation completely defines the domain of definition of the social welfare function U:

$$\begin{aligned} -D \cdot \mathbf{p} &\leqslant \mathbf{b}, \\ -\mathbf{p} &\leqslant 0. \end{aligned} \tag{9}$$

According to standard reference [11], a nonempty intersection of half-spaces defines a set called *polyhedron.* Moreover, a bounded polyhedron is also known as *polytope*.

Proposition 1. The domain of definition of social welfare function U defined by (9) is a polytope.

Proof: Observe that (9) represents a finite intersection of half-spaces, thus defining a polyhedron. Moreover, we can use the positivity conditions of $\mathbf{p}$ to construct a lower bound. Moving up in the intermediation DAG, starting from its bottom layer denoting buyers (see example from Fig. 1), a set of upper bounds, one for each transaction price, can be recursively defined. Thus, we obtain that the domain of definition of social welfare function U is in fact a bounded polyhedron, i.e., a polytope. □

Let us now focus on studying the convexity of function U. We first compute the first and second order derivatives of U, as follows:

$$\frac{\partial U}{\partial p_{i,j}} = \frac{1}{u_i} - \frac{1}{u_j}, \tag{10}$$

$$\frac{\partial^2 U}{\partial p_{i,j}^2} = -\left(\frac{1}{u_i^2} + \frac{1}{u_j^2}\right),$$

$$\frac{\partial^2 U}{\partial p_{i,j}\partial p_{i,k}} = \frac{\partial^2 U}{\partial p_{j,i}\partial p_{k,i}} = -\frac{1}{u_i^2},$$

$$\frac{\partial^2 U}{\partial p_{i,j}\partial p_{j,k}} = \frac{\partial^2 U}{\partial p_{j,k}\partial p_{i,j}} = \frac{1}{u_j^2}.$$

Analysing (10) we observe that the Jacobian and Hessian matrices of U are $1 \times e$ and, respectively, $e \times e$ matrices and they can be defined by:

$$J_U(\mathbf{p}) = \left[\frac{1}{\mathbf{u}}\right]^{\mathrm{T}} D, \tag{11}$$

$$H_U(\mathbf{p}) = D^{\mathrm{T}}\mathrm{diag}\left(\left[-\frac{1}{\mathbf{u}^2}\right]\right) D.$$

The Hessian matrix H_U is used for checking that the social welfare function U is concave, while both Hessian and Jacobian matrices H_U and J_U are also used to implement the nonlinear convex optimization problem.

Proposition 2. Concavity of social welfare function. The Nash social welfare function of a collectively profitable DAG $\mathcal{G}$ defined by (5) on the domain defined by (9) is a concave function.

Proof: The proof utilizes an adaptation of Sylvester's criterion for semi-definite positive or negative matrices. Basically, in our case we must prove that the Hessian matrix $H_U(\mathbf{p})$, is negative semi-definite. This result is obtained by checking that all its principal minors of order k have sign $(-1)^k$. According to [10], this condition entails that function U is concave.

Let $\Delta_{i_1,i_2,\ldots,i_k}$ be a principal minor of the Hessian matrix $H_U(\mathbf{p})$ such that $1 \leq i_1 < i_2 < \cdots < i_k \leq e$ denote k arbitrary arcs of $\mathcal{G}$. Let $\mathcal{G}'$ be the undirected sub-graph of $\mathcal{G}$ induced by the subset $\{i_1, i_2, \ldots, i_k\}$ of arcs and then by discarding its arc orientations. We denote the set of vertices of $\mathcal{G}'$ with V'.

In the case that $\mathcal{G}'$ does contain cycles we can easily observe that $\Delta_{i_1,i_2,\ldots,i_k} = 0$. Let us consider a cycle together with its orientation. This induces the orientation to each of its component arcs. Now if we determine the algebraic sum of the Hessian columns that correspond to all the arcs of the cycle, taking signs according to the arc orientation relative to the cycle orientation, we obviously obtain zero. Note that the

Hessian columns correspond exactly to the arcs of the cycle, so pairs of terms of this sum will cancel each other, thus producing the result zero.

In the case that $\mathcal{G}'$ does not contain cycles we consider the connected components $\mathcal{G}'_j$ of $\mathcal{G}'$, $1 \leq j \leq p$, such that $\mathcal{G}'_j$ contains k_j arcs, $\sum_{j=1}^{p} k_j = k$, and $\mathcal{G}'_j$ has vertices V'_j. Then, minor $\Delta_{i_1,i_2,\ldots,i_k}$ is defined by the next equation:

$$\Delta_{i_1,i_2,\ldots,i_k} = (-1)^k \prod_{j=1}^{p} \sum_{S_j \subseteq V'_j, |S_j|=k_j} \prod_{i \in S_j} \frac{1}{u_i^2} \,. \tag{12}$$

Equation (12) clearly shows that $(-1)^k \Delta_{i_1,i_2,\ldots,i_k} > 0$. □

Theorem 1. Existence and uniqueness of optimal pricing strategy. For each collectively profitable DAG $\mathcal{G}$ there exists a unique optimal pricing strategy that participant agents can collectively apply to maximize the Nash social welfare of $\mathcal{G}$.

Proof: Following Proposition 2 we can conclude that social welfare function U of $\mathcal{G}$ is concave. According to Proposition 1, U is well defined on a polytope domain. Moreover, each polytope is a bounded convex set, so U is in fact a concave function defined on a bounded convex set. We can conclude now that U has a local maximum that is also a global maximum. This proves the existence and uniqueness of an optimal pricing strategy that participant agents can collectively apply to maximize their Nash social welfare. □

4. Case study

In this section we consider the detailed example of the sample intermediation graph presented in Fig. 1. For this example we are going to present its associated matrices, vectors, as well as the explanation of the proof of Proposition 2.

Matrices **b** and D for the sample intermediation DAG $\mathcal{G}$ shown in Fig. 1 are introduced by equations (13) and (14). Note that in this case there are $n = 8$ vertices, i.e., participant agents, as well as $e = 8$ arcs, i.e., transactions.

$$\mathbf{b} = \begin{bmatrix} -\sigma_1 \\ -\sigma_2 \\ 0 \\ 0 \\ \beta_5 \\ 0 \\ \beta_7 \\ \beta_8 \end{bmatrix}, \tag{13}$$

$$D = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}. \tag{14}$$

Each column of the incidence matrix D represents a unique arc of $\mathcal{G}$. For example, arcs 1, 3, and 4 corresponding to columns 1, 3, and 4 of D are as follows: $(1,3)$, $(2,4)$, and $(2,5)$.

Moreover, Jacobian and Hessian matrices of the sample intermediation DAG shown in Fig. 1 are given by (15) and (16).

(15) $J_U(\mathbf{p}) = \left[\frac{1}{u_1} - \frac{1}{u_3} \quad \frac{1}{u_1} - \frac{1}{u_4} \quad \frac{1}{u_2} - \frac{1}{u_4} \quad \frac{1}{u_2} - \frac{1}{u_5} \quad \frac{1}{u_3} - \frac{1}{u_6} \quad \frac{1}{u_4} - \frac{1}{u_6} \quad \frac{1}{u_6} - \frac{1}{u_7} \quad \frac{1}{u_6} - \frac{1}{u_8}\right]$,

(16)
$$H_U(\mathbf{p}) = \begin{bmatrix} -\frac{1}{u_1^2} - \frac{1}{u_3^2} & -\frac{1}{u_1^2} & 0 & 0 & \frac{1}{u_3^2} & 0 & 0 & 0 \\ -\frac{1}{u_1^2} & -\frac{1}{u_1^2} - \frac{1}{u_4^2} & -\frac{1}{u_4^2} & 0 & 0 & \frac{1}{u_4^2} & 0 & 0 \\ 0 & -\frac{1}{u_4^2} & -\frac{1}{u_2^2} - \frac{1}{u_4^2} & -\frac{1}{u_2^2} & 0 & \frac{1}{u_4^2} & 0 & 0 \\ 0 & 0 & -\frac{1}{u_2^2} & -\frac{1}{u_2^2} - \frac{1}{u_5^2} & 0 & 0 & 0 & 0 \\ \frac{1}{u_3^2} & 0 & 0 & 0 & -\frac{1}{u_3^2} - \frac{1}{u_6^2} & -\frac{1}{u_6^2} & \frac{1}{u_6^2} & 0 \\ 0 & \frac{1}{u_4^2} & \frac{1}{u_4^2} & 0 & -\frac{1}{u_6^2} & -\frac{1}{u_4^2} - \frac{1}{u_6^2} & \frac{1}{u_6^2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{u_6^2} & \frac{1}{u_6^2} & -\frac{1}{u_6^2} - \frac{1}{u_7^2} & \frac{1}{u_7^2} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{u_7^2} & -\frac{1}{u_7^2} - \frac{1}{u_8^2} \end{bmatrix}.$$

Let us now consider the following cycle of $\mathcal{G}$: (1, 3), (3, 6), (6, 4), (4, 1). Representing arcs by their indices and assigning one of signs $+$ and $-$ to each arc following its orientation relative to the cycle orientation, we get the cycle: $1,\ 5,\ -6,\ -2$. Now it should be rather easy to verify that adding columns 1 and 5 and then subtracting columns 2 and 6 of matrix H_U, zero is obtained, i.e., $\Delta_{1,5,6,2} = 0$.

Let us now consider the minor $\Delta_{1,3,4}$ corresponding to a sub-graph of $\mathcal{G}$ that does not contain cycles. In this case our sub-graph $\mathcal{G}'$ has 5 vertices $\{1, 2, 3, 4, 5\}$ and $p = 2$ connected components (for the meaning of p please refer to (12)). Moreover, as there are 3 arcs we have $k = 3$. Connected components of sub-graph $\mathcal{G}'$ are defined by subsets of vertices $\{1, 3\}$ and $\{2, 4, 5\}$ so in this case $k_1 = 1$ and $k_2 = 2$. So $\Delta_{1,3,4}$ is defined by

(17) $$\Delta_{1,3,4} = -\left(\frac{1}{u_1^2} + \frac{1}{u_3^2}\right) \cdot \left(\frac{1}{u_2^2 \cdot u_4^2} + \frac{1}{u_2^2 \cdot u_5^2} + \frac{1}{u_4^2 \cdot u_5^2}\right),$$

and obviously $(-1)^3 \cdot \Delta_{1,3,4} > 0$.

5. Computational experiments

5.1. Experimental setup

In this section we present the computational experiments that we performed for assessing the correctness and feasibility of our proposed approach. There are many optimization packages, including more general and powerful frameworks, as well as specialized language-specific libraries with customized APIs.

In this work we have employed the CVXOPT software package for convex optimization [3]. The software prototype was implemented using the 64-bit (AMD64) version of Python 3.7.3 on an ×64-based PC with a 2 cores / 4 threads Intel© Core™ i7-5500U CPU at 2.40 GHz and running Windows 10.

Firstly we mapped our problem to the general framework of nonlinear convex optimization required by CVXOPT package. For this purpose we define two matrices G of size $(e + n) \times e$ and $\mathbf{h}$ of size $(n + e) \times 1$ using the equations:

(18) $$G = \begin{bmatrix} -D \\ -I_e \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} \mathbf{b} \\ 0 \end{bmatrix}.$$

Our optimization problem, captured as nonlinear convex optimization problem suitable for CVXOPT implementation, can be represented using

(19) $$\begin{aligned} \text{minimize} \quad & -U(\mathbf{p}) = -\sum_{i=1}^{n} \log(D \cdot \mathbf{p} + \mathbf{b})_i, \\ \text{subject to} \quad & G \cdot \mathbf{p} \preccurlyeq \mathbf{h}. \end{aligned}$$

Our implementation using CVXOPT was based on the cvxopt.solvers.cp solver. In the implementation process we have followed the sequence of steps [3]:

1. Define a Python function, let us call it `U`, to evaluate the optimization objective, as well as to check the optimization constraints.

2. Implement Python function `U` using the definitions of the Jacobian and Hessian matrices of the objective function, according to (11). Their implementation has been done using the CVXOPT-specific data type cvxopt.matrix.

3. Define a point `x0` inside the domain of the objective function that is used by function `U`. This point has bencomputed before calling the solver, following the details shown below.

4. Prepare a collection of data sets representing problem instances. Each data set captures a single problem instance corresponding to a specific intermediation DAG and its associated parameters (limit prices). More details regarding the data sets are given below.

5. Define a main Python script to describe the whole experiment. This script loads the data sets, configures the CVXOPT solver with suitable parameters, calls the solver function cvxopt.solvers.cp, and then obtains the solution representing the optimization outcome.

We experimentally analysed two possible and distinct options to compute the required point inside the polytope domain:

1. Map the polytope half-space representation (as given by (9)) to the equivalent vertex representation [11], and then determine the point by sampling the polytope interior with a uniform probability distribution.

2. Directly determine the point using the given half-space representation (9). For that purpose we have used the Chebyshev center of the polytope that represents the deepest point inside the polytope [7].

We experimentally have tested both approaches on sample polytopes, employing the PYPOMAN software package for polyhedral manipulations [8]. Our conclusion has revealed that the first approach is not usable. This is caused by the excessively large running time of the vertex computation, most probably because the resulting equivalent polytopes could generally have a too large number of vertices. For example, for one example intermediation DAG comprising 15 vertices and 35 arcs (i.e., 35 dimensions of the polytope representation), the resulting vertex representation of this polytope had 58,795 vertices.

Based on these conclusions, we have chosen the second method to complete our experimental workflow. Nevertheless, it was useful to implement two different methods for generating points inside the polytope, at least for smaller problem sizes,

to be able to analyse the method convergence for distinct initial points inside the polytope.

5.2. Data sets

We generated an artificial data set comprising many random DAGs of different sizes, capturing different intermediation networks. The generating process was parameterized as follows:

1. The number n of graph vertices, defined as an element of the following set of possible values: $\{5, 6, 7, 8, 9, 10, 15, 20, 30, 40, 50\}$.

2. The density factor *f* of the DAG. The higher is the value of the density factor the more arcs our graph contains. Value of *f* is given as the percentage of the number of arcs, from the total possible number of arcs of the DAG, i.e., $\frac{n\cdot(n-1)}{2}$, defined as an element of the set $\{10, 20, 30, 40, 50, 60, 70, 80, 90\}$.

3. Number ng of DAGs for each pair of values *n* and *f*. In our experiments we have set $\text{ng} = 10$.

Using these values, we obtained a total number of $11 \cdot 9 \cdot 10 = 990$ DAGs in our generated data set. For the representation of each DAG we have used its adjacency matrix A defined according to (20). Then we converted it into the incidence matrix representation D, that was used for the optimization goal:

(20) $$A_{i,j} = \begin{cases} 1 & \text{if there exists an arc from } i \text{ to } j, \\ 0 & \text{otherwise.} \end{cases}$$

For each DAG we stored its adjacency matrix into a separate text file. The name of this file was defined to easily identify the parameters of the generating process. For example, the 5th DAG consisting of 15 vertices and 40% density factor was stored into a text file named graf-15-40-5.txt.

We close this section with two remarks regarding the process of randomly generating DAGs to meet the goals of our experiments:

1. The standard generation of random graphs according to a pre-defined density factor is not enough. The generation process had to ensure that the resulting directed graph is in fact a proper DAG. This desideration was achieved by constraining the generating process to always produce adjacency matrices in upper triangular form. This condition assures that the output directed graph is a DAG. Conversely, this process is not restricting the generated DAGs. The adjacency matrix representation of any possible DAG can be mapped to upper triangular format by performing a renumbering the graph vertices according to one of the topological orderings of the graph.

2. In the generating process we have discarded as being not relevant all the DAGs that contain "singular" nodes, without incoming and outgoing arcs, as these graphs were deemed unsatisfactory to capture intermediation DAGs.

Last but not least, we chose appropriate values for the limit prices of the buyers and sellers. According to our theoretical results from [12], a necessary condition to guarantee collective profitability of an intermediation DAG is

(21) $$\sum_{b\in\mathcal{B}} \beta_b \geq \sum_{s\in\mathcal{S}} \sigma_s.$$

Let $n_b = |\mathcal{B}|$ and $n_s = |\mathcal{S}|$ denote the number of buyers and sellers. We chose $\boldsymbol{\sigma}$ and $\boldsymbol{\beta}$ according to

$$\begin{aligned} \sigma_s &= \boldsymbol{\sigma} = 100 && \text{for all } s \in \mathcal{S}, \\ \beta_b &= \boldsymbol{\beta} = \boldsymbol{\sigma} \cdot \left(\left\lceil \frac{n_b}{n_s} \right\rceil + 1\right) && \text{for all } b \in \mathcal{B}. \end{aligned} \tag{22}$$

Observe that this choice guarantees that condition (21) holds. However condition (22) is only necessary, but not sufficient. So for few data sets, the optimization procedure could fail to obtain a solution simply because such a solution does not exist. This aspect is highlighted in the next section of the paper.

For example, if there are $n_b = 3$ buyers and $n_s = 2$ sellers, (22) gives us a limit price of 100 for each seller, as well as a limit price of 300 for each buyer.

5.3. Results and discussion

The information regarding the size of our data set is summarized in Table 1. We present the minimum and maximum number of arcs for each DAG from the subset of DAGs characterized by the same number of nodes. Additionally, we have compared the maximum number of arcs of a DAG from our data set, with the maximum number of arcs of a fully connected DAG with n of nodes, i.e., $\frac{n\cdot(n-1)}{2}$. This value can be obtained for DAGs such that their adjacency matrix contains only ones in its upper triangle, while the lower triangle and the main diagonal contain only zeroes.

Table 1. Size of data sets

Number of nodes	Number of arcs		
n	min (data set)	max (data set)	max $\frac{n\cdot(n-1)}{2}$
5	3	10	10
6	3	15	15
7	4	21	21
8	4	28	28
9	5	36	36
10	7	45	45
15	12	99	105
20	19	177	190
30	39	401	435
40	67	721	780
50	107	1111	1225

Analysing Table 1, we see that when the number of vertices is smaller, i.e., $n \in \{5, 6, 7, 8, 9, 10\}$ our data set contains fully connected DAGs. Nevertheless, for larger values of n this does not hold. For example, analysing the graphs with $n = 40$ vertices, we observe that the maximum number of arcs of the DAGs in our data set is 721, while a fully connected DAG with 40 vertices has 780 arcs.

The number of vertices and arcs of the intermediation DAG provides an estimate of the "size" of the optimization problem. Parameter n represents the number of participant agents. It also gives the number of terms of the sum defining the Nash social welfare function (see (5)). Parameter e represents the number of transactions. It also equals the number of decision variables of the optimization problem. Our largest optimization problem had 1111 decision variables, as Table 1 clearly

illustrates. Note however that the largest optimization problem that we could successfully solve was graph-50-90-2 comprising 1102 transactions.

We have created a Python script for invoking the convex optimization engine on each problem instance of our data set. We configured the solver as follows:

1. The maximum number of iterations, captured by **`maxiters`** parameter, was set to **`30`**.

2. The number of iterative refinement steps when solving Karush–Kuhn–Tucker equations, captured by **`refinement`** parameter, was set to **`2`**.

3. The flag for turning on the output of the optimization progress, captured by **`show_progress`** parameter, was set to **`True`**.

4. The various tolerances, captured by **`abstol`**, **`reltol`**, and **`feastol`** parameters, were initialized with their default values **`1×10⁻⁷`**, **`1×10⁻⁶`**, and respectively **`1×10⁻⁷`**.

The maximum number of iterations that were required for successfully solving an optimization problem was 29. This was obtained for data set graph-40-10-8.

We recorded the computation times spent by running the optimization process, for each set of problem instances with a given number of participants. These results are presented in Table 2.

It is worth observing that the total running time required for solving all the problems with 50 vertices, was significantly larger than for all the other problems. This is explained by the fact that convergence failed for a total of 13 problems, all of them representing DAGs with 50 vertices.

Table 2. Computation time

Number of nodes n	Time, s
5	1.875
6	1.875
7	1.953
8	2.062
9	2.906
10	3.437
15	6.390
20	9.890
30	38.843
40	134.109
50	1589.890
Total	1782.230

We make one important note before presenting our results of the optimization process. As it is already stated in our previous work [12], the sum of utilities of all the participants of an intermediation network is constant, as shown by:

$$\sum_{i=1}^{n} u_i = \sum_{b\in\mathcal{B}} \beta_b - \sum_{s\in\mathcal{S}} \sigma_s \,. \tag{23}$$

Therefore, if we denote by u_δ the increment utility determined using (24), and if we apply the classic inequality between arithmetic and geometric means, we are able to obtain the upper bound of Nash social utility function, as given by (25):

$$u_\delta = \frac{(\sum_{b\in\mathcal{B}} \beta_b - \sum_{s\in\mathcal{S}} \sigma_s)}{n}, \tag{24}$$

$$U(\mathbf{p}) \le n \log u_\delta. \tag{25}$$

Note that the upper bound $n \log u_\delta$ of Nash social utility function can be achieved if and only if linear system

(26) $$(D \cdot \mathbf{p} + \mathbf{b})_i = u_\delta \quad \text{for all} \quad i = 1, \dots, n,$$

has positive solutions.

We can use this observation to distinguish between two different situations when our solver will return an optimal solution:

1. The optimal solution of our problem is obtained when all participants receive the same utility. In this case solving linear system (26) will produce positive values representing feasible transaction prices that determine equal utilities for all participants. For obvious reasons, we will label this case as "trivial" in what follows.

2. If linear system (26) does not have positive solutions then we must run the optimization solver to obtain a solution that maximizes that Nash social welfare. Obviously, in this case the utilities of participants will not be equal. For obvious reasons, we will label this case as "non-trivial" in what follows.

There are also two different situations when the optimization solver fails to obtain solutions:

1. The optimization terminates after performing the maximum number of iterations, without reaching convergence. We label this situation as "unknown", with the obvious meaning that we do not know the optimality status of the obtained solution.

2. The optimization terminates abruptly with the solver raising software exception. We label this situation as "exception". One possible explanation could be that in this particular case the intermediation DAG is not collectively profitable. Recall that data sets were generated by assuring only the necessary condition of collective profitability. Therefore it is possible that some of our problem instances represent intermediation DAGs that are not collectively profitable.

The optimization results are shown in Table 3. For each set of graphs with the same given number of nodes we distinguish between each of the four possible optimization outcomes: "trivial", "non-trivial", "exception", and "unknown". Unsurprisingly, in most of the situations, the optimization engine produced the trivial solution. Nevertheless, for a rather significant number of cases, i.e., between 17.5% (for $n = 40$) and 44.2% (for $n = 7$), a non-trivial solution was obtained. These results clearly depend on the problem structure (i.e., the underlying DAG), as well as on the values of limit price vectors $\boldsymbol{\sigma}$ and $\boldsymbol{\beta}$. For example, a non-trivial solution was obtained for problem graph-50-20-9. The problem has 50 vertices, 216 arcs, while the convergence was obtained in 22 iterations.

We also observed that the optimization engine terminated for at least one case by raising software exception (most probably because the intermediation DAG was not collectively profitable), for all the values of the number of participants n from 7 to 50. A deeper investigation revealed that in all these cases (20 in total, see the column of Table 3 titled *Exception*), the DAG density factor was at most 40%, while in 17 of these cases the density factor of the graph was at most 20%, i.e., the graph was sparse.

Last but not least observe that the optimization solver terminated neither with exception nor with convergence (see the column of Table 3 titled *Unknown*) for 13

problems of 50 vertices. We believe that these outcomes were either caused by the internal functionality of the solver or by the fact that the underlying DAG is not collectively profitable. It is interesting to note that this happened for DAGs with density factor above 70%, i.e., for rather dense graphs,

Table 3. Optimization results

Number of nodes	Success		Failure	
n	Trivial	Non-trivial	Exception	Unknown
5	68	22	0	0
6	68	22	0	0
7	61	27	2	0
8	72	15	3	0
9	65	20	5	0
10	68	21	1	0
15	72	14	4	0
20	61	27	2	0
30	80	9	1	0
40	82	7	1	0
50	67	9	1	13

6. Conclusion

Our main achievement is the formulation of the problem of determining optimal pricing strategies in semi-competitive intermediation networks, as nonlinear convex optimization. The optimization criterion targets maximizing Nash social welfare of the whole society of participant agents to the intermediation network. We obtained theoretical results stating that if the network is collectively profitable then there always exists a globally optimal pricing strategy that participants can employ to maximize their Nash social welfare. Additionally we have presented evidence of the feasibility of our approach by running computational experiments using a nonlinear convex optimization package. The experimental results are consistent with our theoretical conclusions and they support the practical value of our proposed model.

References

1. Ahuja, R. K., T. L. Magnanti, J. B. Orlin. Network Flows: Theory, Algorithms, and Applications. Pearson, 1993.
2. Andersen, M., J. Dahl, Z. Liu, L. Vandenberghe. Interior-Point Methods for Large-Scale Cone Programming. – In: S. Sra, S. Nowozin, S. J. Wright, Eds. Optimization for Machine Learning, MIT Press, 2011, pp. 1-26.
3. Andersen, M., J. Dahl, L. Vandenberghe. CVXOPT User's Guide. Release 1.2.4, 20 January 2020.
   https://cvxopt.org/userguide
4. Bădică, A., C. Bădică, M. Ivanović, I. Buligiu. Collective Profitability and Welfare in Selling-Buying Intermediation Processes. – In: N. Nguyen, L. Iliadis, Y. Manolopoulos, B. Trawiński, Eds. Computational Collective Intelligence. ICCCI 2016. Part II, Lecture Notes in Computer Science. Vol. **9876**. Cham, Springer, 2016, pp. 14-24.
5. Bădică, A., C. Bădică, M. Ivanović, D. Logofătu. Collective Profitability of DAG-Based Selling-Buying Intermediation Processes. – In: J. Del Ser, E. Osaba, M. N. Bilbao, J. J. Sánchez-Medina, M. Vecchio, X.-S. Yang, Eds. Intelligent Distributed Computing XII. Studies in Computational Intelligence. Vol. **798**. Cham, Springer, 2018, pp. 414-424.

6. B ă d i c ă, C., M. G a n z h a, M. P a p r z y c k i, A. P î r v ă n e s c u. Experimenting with a Multi-Agent E-Commerce Environment. – In: V. Malyshkin V, Ed. Parallel Computing Technologies. PaCT 2005. Lecture Notes in Computer Science. Vol. **3606**. Berlin, Heidelberg, Springer, 2005, pp. 393-402.
7. B o y d, S., L. V a n d e n b e r g h e. Convex Optimization. Cambridge University Press, 2004.
8. C a r o n , S. PYthon Module for POlyhedral MANipulations – PYPOMAN. Version 1.0. 2020. **https://scaron.info/doc/pypoman/**
9. K a n e k o, M., K. N a k a m u r a. The Nash Social Welfare Function. – Econometrica, Vol. **47**, 1974, No 2, pp. 423-435.
10. P r u s s i n g, J. E. The Principal Minor Test for Semidefinite Matrices. – Journal of Guidance, Control, and Dynamics, Vol. **9**, 1986, No 1, pp. 121-122.
11. T h o m a s, R. R. Lectures in Geometric Combinatorics. – In: Student Mathematical Library: IAS/Park City Mathematical Subseries. Vol. **33**. American Mathematical Society, 2006.
12. B ă d i c ă, A., C. B ă d i c ă, M. I v a n o v i ć, D. L o g o f ă t u. Collective Profitability in Semi-Competitive Intermediation Networks. – Journal of Intelligent and Fuzzy Systems, Vol. **37**, 2019, pp. 7357-7368.
13. P a r s o n s, S., J. A. R o d r í g u e z-A g u i l a r, M. K l e i n. Auctions and Bidding: A Guide for Computer Scientists. – ACM Computing Surveys, Vol. **43**, 2011, No 2, pp. 10:1-10:59.
14. L a u d o n, K. C., C. G. T r a v e r. e-Commerce 2019: Business, Technology and Society. 15th Edition. Pearson, 2019.

Received: 06.11.2019; Second Version: 05.03.2020; Third Version: 17.04.2020; Accepted: 22.04.2020