Network-based Receivable Financing

I. Bordino F. Gullo

UniCredit
R&D Department
Rome, Italy
{ilaria.bordino, francesco.gullo}@unicredit.eu

The 27th ACM International Conference
on Information and Knowledge Management (CIKM 2018)
October 22-26, 2018
Turin, Italy
Application scenario:
Traditional (client-server) receivable financing

- A **receivable** is a debt owed to a company by its customers for goods or services that have been delivered or used but not yet paid for
  - e.g., **invoices**

- **Receivable Financing (RF)** is a service for creditors to fund cash flow by selling accounts receivables to a **funder** or **financing company**
  - Benefits for funder: **service fee**
  - Benefits for customers: **instant access to capital, no credit control**

- Existing funders adopt a **client-server** approach
  - Each request for a receivable to be funded is **handled individually** by the funder
A novel, network-based approach to receivable financing

Major limitation of client-server receivable financing
It disregards the fact that receivables constitute a network where the same customer may act as a creditor or a debtor of different receivables

Proposal
A novel approach to receivable financing where a network perspective is profitably exploited to trigger a money flow among customers themselves

Pros for the funder:
- More liquidity
- Reduced risk of exposure

Pros for customers:
- Smaller fees
- Reduced time and effort in service establishment
A novel, network-based approach to receivable financing

- **Introduction**
- **Service overview**
- **Problem definition**
- **Algorithms**
- **Experiments**

**Background**

**Motivation**

**Challenges and contributions**

**Outline**

A novel, network-based approach to receivable financing

(a) Client-server receivable financing

| Receivable 1 | Creditor: B | Debtor: A | Amount: 600 |
| Receivable 2 | Creditor: B | Debtor: A | Amount: 700 |

(b) Network-based receivable financing

- Receivable 1
  - Creditor: B
  - Debtor: A
  - Amount: 600

- Receivable 2
  - Creditor: B
  - Debtor: A
  - Amount: 700

- Receivable 3
  - Creditor: C
  - Debtor: B
  - Amount: 1000

- Receivable 4
  - Creditor: A
  - Debtor: C
  - Amount: 900

I. Bordino, F. Gullo

Network-based Receivable Financing
Challenges and contributions

Main challenge

Given a network of receivables, identify a proper subset of receivables to be settled (i.e., for which the receivable-financing service is provided)

Contributions

- Formulation of network-based receivable settlement as a novel combinatorial-optimization problem
- Theoretical characterization of that problem
  - \textbf{NP}-hardness, bounds on the objective-function value of a set of solutions
- An exact branch-and-bound algorithm
- A more efficient algorithm
  - based on a relaxation of the original problem, and its theoretical characterization (\textbf{NP}-hardness and connection with \textsc{Knapsack}-like problems)
- A hybrid algorithm, as an ultimate proposal
Outline

- Introduction: motivation, challenges, contributions
- Service overview
- Problem definition
- Algorithms
  - An exact algorithm
  - A more efficient algorithm
  - A hybrid algorithm
- Experiments
Outline

- Introduction: motivation, challenges, contributions
- **Service overview**
- Problem definition
- Algorithms
  - An exact algorithm
  - A more efficient algorithm
  - A hybrid algorithm
- Experiments
A receivable $R \in \mathcal{R}$ is an object with the following attributes:

- $\text{amount}(R) \in \mathbb{R}$: amount of the receivable
- $\text{creditor}(R) \in \mathcal{U}$: customer being the payee of the receivable
- $\text{debtor}(R) \in \mathcal{U}$: customer being the payer of the receivable
- $\text{insertdate}(R)$: date the receivable was added to the system
- $\text{duedate}(R)$: date on which the payment falls due
- $\text{life}(R) \in \mathbb{N}$: the maximum number of days the network-based RF service is allowed to try to settle the receivable

$R$ is said **active** for $\text{creditor}(R)$, and **passive** for $\text{debtor}(R)$
Every **customer** \( u \in \mathcal{U} \) is assigned the following attributes:

- \( bl_r(u) \in \mathbb{R} \): *receivable balance* of \( u \)'s account
- \( bl_a(u) \in \mathbb{R} \): *actual balance* of \( u \)'s account
- \( cap(u) \in \mathbb{R} \): upper bound on the receivable balance of \( u \)'s account
  - requiring \( bl_r(u) \leq cap(u) \) at any time **avoids unbalanced situations** where a customer utilizes the service only to get money without paying passive receivables
- \( fl(u) \in \mathbb{R} \): lower bound on the actual balance of \( u \)'s account
Network-based receivable financing in action

1. Creditor submits a receivable \( R \), setting \( \text{life}(R) \)
2. System asks \( \text{debtor}(R) \) for confirmation
3. \( R \) is added to the set \( \mathcal{R} \) of current receivables
4. System attempts to settle \( R \) during the period
   \([\text{insertdate}(R), \min\{\text{insertdate}(R) + \text{life}(R), \text{duedate}(R)\}]\]
5. If no settlement happens, the receivable is returned to the creditor; otherwise,
   \( \text{amount}(R) \) is transferred from \( \text{debtor}(R) \) to \( \text{creditor}(R) \)

Do-ut-des principle

The debtor is encouraged to accept paying a receivable before its \( \text{duedate} \) to gain operability within the service, so as to get her (future) active receivables settled more easily → due to the constraint \( bl_r(u) \leq cap(u) \)
Outline

- Introduction: motivation, challenges, contributions
- Service overview
- **Problem definition**
- Algorithms
  - An exact algorithm
  - A more efficient algorithm
  - A hybrid algorithm
- Experiments
Input: active receivables and S-multigraph

- Receivable settlement works on a daily basis, running offline at the end of any working day $t$
- Input: set $\mathcal{R}(t)$ of valid receivables at time $t$
- $\mathcal{R}(t)$ describes a directed, weighted, node-attributed multigraph

**Definition (S-multigraph)**

Given a set $\mathcal{R}(t)$ of receivables active at time $t$, the *S-multigraph* induced by $\mathcal{R}(t)$ is a triple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, where $\mathcal{V}$ is a set of nodes, $\mathcal{E}$ is a multiset of ordered pairs of nodes, i.e., arcs, and $w : \mathcal{E} \to \mathbb{R}^+$ is a function assigning (positive real) weights to arcs. Each arc $(u, v) \in \mathcal{E}$ models the case “$u$ pays $v$”, i.e., it corresponds to a receivable $R \in \mathcal{R}(t)$ where $u = \text{debtor}(R)$, $v = \text{creditor}(R)$, and $w(u, v) = \text{amount}(R)$. Each node $v \in \mathcal{V}$ is assigned attributes $bl_r(u)$, $lb_a(u)$, $cap(u)$, and $fl(u)$. 
The Max-profit Balanced Settlement problem

- **Objective**: maximize the total amount of selected receivables desirable for both funder and customers

- **Constraints**:
  1. Consistency with $fl$-cap range: $bl_r(u) \leq cap(u)$, $bl_a(u) \geq fl(u)$
  2. Selected customers should be both payers and payees → strategic marketing choice

**Problem (Max-profit Balanced Settlement)**

Given an $S$-multigraph $G = (V, E, w)$, find a multisubset $E^*$ of arcs so that

$$
E^* = \arg \max_{\hat{E} \subseteq E} \sum_{e \in \hat{E}} w(e) \quad \text{subject to}
$$

$$
\left( \sum_{(v,u) \in \hat{E}} w(v, u) - \sum_{(u,v) \in \hat{E}} w(u, v) \right) \in [fl(u) - bl_a(u), cap(u) - bl_r(u)],
$$

$$
|\{(u, v) | (u, v) \in \hat{E}\}| \geq 1, \text{ and } |\{(v, u) | (v, u) \in \hat{E}\}| \geq 1,
$$

\(\forall u \in V(\hat{E}) = \{u \in V | (u, v) \in \hat{E} \lor (v, u) \in \hat{E}\}\).
The **Max-profit Balanced Settlement** problem

I. Bordino, F. Gullo

Network-based Receivable Financing
Outline

- Introduction: motivation, challenges, contributions
- Service overview
- Problem definition
- Algorithms
  - An exact algorithm
  - A more efficient algorithm
  - A hybrid algorithm
- Experiments
The Settlement-BB algorithm: search space

- **Binary tree** $\mathcal{T}$ with $|\mathcal{E}| + 1$ levels
- **Levels (but the root)** $\equiv$ arcs in $\mathcal{E}$
- **Root-to-leaf paths** $\equiv$ individual solutions $\hat{\mathcal{E}} \in 2^\mathcal{E}$
- **Non-leaf tree node** $\equiv$ set of solutions

*Binary tree $\mathcal{T}$ with $|\mathcal{E}| + 1$ levels*

- Arc 1
  - Arc 2
    - Arc 3
      - Arc 1: “YES”, $\{\text{Arc 1, Arc 2}\}$
      - Arc 3: “NO”
    - Arc 3: “YES”, $\{\text{Arc 1}\}$
  - Arc 2: “NO”, $\{\text{Arc 2}\}$
  - Arc 2: “YES”, $\{\text{Arc 2}\}$
The Settlement-BB algorithm: search-space exploration

Algorithm 1: Settlement-BB

**Input:** An S-multigraph $G = (V, E, w)$  
**Output:** A multiset $E^* \subseteq E$

1: $T :=$ tree-like representation of $2^E$
2: $X \leftarrow \{\text{root of } T\}$, $LB_{\text{max}} \leftarrow 0$
3: **while** $X$ contains some non-leaf tree-nodes **do**
4: $X \leftarrow$ extract (and remove) a non-leaf tree-node from $X$
5: $UB_X \leftarrow$ upper bound on the solutions spanned by $X$ \{Alg. 3\}
6: **if** $UB_X \geq LB_{\text{max}}$ **then**
7: $LB_X \leftarrow$ lower bound on the solutions spanned by $X$ \{Alg. 2\}
8: **if** $LB_X = UB_X$ **then** $E^* \leftarrow arcs(X)$ and stop the algorithm
9: $LB_{\text{max}} \leftarrow \max\{LB_{\text{max}}, LB_X\}$
10: add all $X$’s children to $X$
11: $L \leftarrow \{\text{leaf } X \in X | arcs(X) \text{ satisfy constraints of Problem 1}\}$
12: $E^* \leftarrow \arg \max_{arcs(X): X \in L} \sum_{e \in arcs(X)} w(e)$

- Standard branch-and-bound exploration
- **Crucial point:** definition of lower bound and upper bound
The Settlement-BB algorithm: lower bound

- For a tree-node $X$ at level $i$ of $T$, 
  \[ E_X = E_X^+ \cup E_X^- : \text{arcs for which a decision has been taken} \]

- **Lower bound** on the solutions spanned by $X$: any feasible solution $\hat{E}$ to MAX-PROFIT BALANCED SETTLEMENT, subject to the additional constraint of containing all arcs in $E_X^+$ and no arcs in $E_X^-$

- Find the set $C$ of cycles of the multigraph induced by $E \setminus E_X^-$

- Greedily selects cycles based on their amount, as long as they meet the $fl$-$cap$ problem constraints
  - other problem constraint always satisfied
  - cycle enumeration is a well-established problem (we use the classic Johnson’s algorithm)

- **Time complexity**
  - dominated by cycle enumeration $\Rightarrow$ we look for cycle up to length $L$
  - the rest takes $O(L|C| \log |C|)$ time
**The Settlement-BB algorithm: lower-bound**

**Algorithm 2: Settlement-BB-LB**

**Input:** An S-multigraph $G = (V, E, w)$, two multisets $E^+_X \subseteq E$, $E^-_X \subseteq E$

**Output:** A multiset $\hat{E} \subseteq E \setminus E^-_X$

1: $C \leftarrow$ cycles of multigraph $G^- = (V, E \setminus E^-_X, w)$ \hspace{2cm} \{Johnson’s algorithm\}

2: $\hat{E} \leftarrow \emptyset$, $\hat{\hat{E}} \leftarrow \emptyset$

3: while $C \neq \emptyset$ \& $E^+_X \notin \hat{E}$ do

4: $C \leftarrow \{C \in C \mid \hat{E} \cup C$ meets Constraint (1) of Max-profit Balanced Settlement\}

5: $C \leftarrow$ cycle in $C$ minimizing $|C \cap (E^+_X \setminus \hat{E})| \times \sum_{e \in C \setminus \hat{E}} w(e)^{-1}$

6: $\hat{E} \leftarrow \hat{E} \cup \{C\}$, $C \leftarrow C \setminus \{C\}$, $\hat{\hat{E}} \leftarrow \hat{E} \cup C$

7: while $C \neq \emptyset$ do

8: $C \leftarrow \{C \in C \mid \hat{E} \cup C$ meets Constraint (1) of Max-profit Balanced Settlement\}

9: $C \leftarrow$ cycle in $C$ maximizing $\sum_{e \in C \setminus \hat{E}} w(e)$

10: $C \leftarrow C \setminus \{C\}$, $\hat{E} \leftarrow \hat{E} \cup C$

11: if $E^+_X \notin \hat{E}$ then $\hat{E} \leftarrow \emptyset$
Relaxation of **Max-profit Balanced Settlement** where

- Constraint (2) is discarded
- Arcs are allowed to be selected **fractionally**

**Problem (Relaxed Settlement)**

Given an S-multigraph $G = (\mathcal{V}, \mathcal{E}, w)$, find

$$\{x_e \in [0, 1]\}_{e \in \mathcal{E}}$$

so as to

Maximize

$$\sum_{e \in \mathcal{E}} x_e w(e)$$

subject to

$$\left( \sum_{e = (v, u) \in \mathcal{E}} x_e w(e) - \sum_{e = (u, v) \in \mathcal{E}} x_e w(e) \right) \in [fl(u) - bl_a(u), cap(u) - bl_r(u)], \ \forall u \in \mathcal{V}$$

The desired upper bound relies on an interesting characterization of the **Relaxed Settlement** problem as a **network-flow problem**:

**Solving Relaxed Settlement on multigraph $G$ is equivalent to solving Min-Cost Flow on a modified version of $G$**
The Settlement-BB algorithm: upper bound

**Algorithm 3: Settlement-BB-UB**

**Input:** An S-multigraph $G = (V, E, w)$, two multisets $E^+_X \subseteq E$, $E^-_X \subseteq E$

**Output:** A real number $UB_X$

1. $G^- := (V, E \setminus E^-_X, w)$
2. $UB_X \leftarrow$ solve Min-Cost Flow applying Theorem 4.2 on $G^-$ and forcing flow $f(e) = w(e)$, $\forall e \in E^+_X$; return $-1$ if no admissible solution exists

  - $\mathcal{O}(|E| (|V| \log |V|) \log(|V| w_{max}))$ time complexity, where $w_{max} = \max_{e \in E} w(e)$
The modified version of \( G \) considered in this context is as follows:

**Definition (S-flow graph)**

The *S-flow graph* \( G_f = (\mathcal{V}_f, \mathcal{E}_f, w_f) \) of an S-multigraph \( G = (\mathcal{V}, \mathcal{E}, w) \) is a simple weighted directed graph where:

- All arcs \((u, v) \in \mathcal{E}\) between the same pair of nodes are collapsed into a single one, and the weight \( w_f(u, v) \) is set to \( \sum_{(u, v) \in \mathcal{E}} w(u, v) \);

- \( \mathcal{V}_f = \mathcal{V} \cup \{\tilde{s}, \tilde{t}\} \), i.e., the node set of \( G_f \) is composed of all nodes of \( G \) along with two dummy nodes \( \tilde{s} \) and \( \tilde{t} \);

- \( \mathcal{E}_f = \mathcal{E} \cup \{(\tilde{s}, u) \mid u \in \mathcal{V}\} \cup \{(u, \tilde{t}) \mid u \in \mathcal{V}\} \cup \{((\tilde{t}, \tilde{s})\}, \) i.e., the arc set of \( G_f \) is composed of (i) all (collapsed) arcs of \( G \), (ii) for each node \( u \in \mathcal{V} \), a dummy arc \((\tilde{s}, u)\) with weight \( w_f(\tilde{s}, u) = bl_a(u) - fl(u) \) and a dummy arc \((u, \tilde{t})\) with weight \( w_f(u, \tilde{t}) = cap(u) - bl_r(u) \), and (iii) a dummy arc \((\tilde{t}, \tilde{s})\) with weight \( w_f(\tilde{t}, \tilde{s}) = \infty \).
The Settlement-BB algorithm: upper bound

**Theorem 4.2**

Given an S-multigraph $G = (\mathcal{V}, \mathcal{E}, w)$, let $G_f = (\mathcal{V}_f, \mathcal{E}_f, w_f)$ be the S-flow graph of $G$. Let also cost, lower-bound, upper-bound and supply/demand functions $c : \mathcal{E}_f \to \mathbb{R}$, $\lambda : \mathcal{E}_f \to \mathbb{R}$, $\mu : \mathcal{E}_f \to \mathbb{R}$ and $b : \mathcal{V}_f \to \mathbb{R}$ be defined as:

- $\lambda(e) = 0$, $\mu(e) = w_f$, $\forall e \in \mathcal{E}_f$;
- $c(\tilde{t}, \tilde{s}) = 0$, and $c(\tilde{s}, u) = c(u, \tilde{t}) = 0$, $\forall u \in \mathcal{V}_f$;
- $c(e) = -1$, $\forall e \in \mathcal{E}_f \cap \mathcal{E}$;
- $b(u) = 0$, $\forall u \in \mathcal{V}_f$.

It holds that solving **Min-Cost Flow** on input $\langle G_f, c, \lambda, \mu, b \rangle$ is equivalent to solving **Relaxed Settlement** on input $G$.

**Corollary**

*Given an S-multigraph $G$, the solution to **Max-profit Balanced Settlement** on $G$ is upper-bounded by the solution to **Min-Cost Flow** on the input $\langle G_f, c, \lambda, \mu, b \rangle$ of Theorem 4.2.*
Outline

- Introduction: motivation, challenges, contributions
- Service overview
- Problem definition
- Algorithms
  - An exact algorithm
  - A more efficient algorithm
  - A hybrid algorithm
- Experiments
The Settlement-\textbf{BEAM} algorithm

- **Main idea**: enumerating cycles and properly selecting a subset

- We formulate the **Optimal Cycle Selection** problem
  - find a subset of cycles exhibiting the maximum total amount and satisfying the constraints of **Max-profit Balanced Settlement**

- We theoretically characterize **Optimal Cycle Selection**
  - **NP**-hardness (reduction from **Maximum Independent Set**)
  - connection with **Knapsack-like problems** (i.e., **Set Union Knapsack**)

- We devise our **Settlement-\textbf{BEAM}** inspired by the well-established Aruselvan’s algorithm for **Set Union Knapsack**
  - arcs $\equiv$ elements, cycles $\equiv$ items
  - extension to handle **Multidimensional Set Union Knapsack**
  - coupling it with a **beam-search** methodology
The Settlement-BEAM algorithm

**Algorithm 4: Settlement-BEAM**

**Input:** An S-multigraph $G = (V, E, w)$, an integer $K$

**Output:** A multiset $E^* \subseteq E$

1: $E^* \leftarrow \emptyset$
2: $C \leftarrow$ cycles of $G$
3: while $C \neq \emptyset$ do
4: $C' \leftarrow K$-sized subset of $C$ by Greedy Max Cover
5: $C_2' \leftarrow \{ \{C_i, C_j\} | C_i, C_j \in C'\}$
6: for all $\{C_i, C_j\} \in C_2'$ do
7: $C_{ij} \leftarrow C_i \cup C_j$
8: process all $C \in C' \setminus \{C_i, C_j\}$ one by one, by non-increasing $\omega(\cdot)$ score (Eq. (3));
9: add $C$ to $C_{ij}$ if $C_{ij} \cup C \cup E^*$ is feasible for Optimal Cycle Selection
10: $E^* \leftarrow E^* \cup \arg \max_{C_{ij} \in C_2'} \sum_{e \in C_{ij}} w(e)$
11: $C \leftarrow C \setminus (C' \cup \{C \in C | C \cap E^* = C\})$

**ω(C) =** \[ \frac{\sum_{e \in C} w(e)}{\sum_{e \in C} \frac{w(e)}{f(e)}} \], where $f(e) = |\{C \in C : e \in C\}|$  \hspace{1cm} (3)

$O(L K^2 |C|)$ time complexity
Outline

- Introduction: motivation, challenges, contributions
- Service overview
- Problem definition
- **Algorithms**
  - An exact algorithm
  - A more efficient algorithm
  - A hybrid algorithm
- Experiments
The Settlement-HYBRID algorithm

- Run Settlement-BB on the smaller connected components
- Run Settlement-BEAM on the larger connected components

### Algorithm 5: Settlement-HYBRID

**Input:** An S-multigraph $G = (V, E, w)$, two integers $H, K$  
**Output:** A multiset $E^* \subseteq E$

1: $E^* \leftarrow \emptyset$, $\text{CC} \leftarrow$ weakly connected components of $G$
2: for all $G \in \text{CC}$ s.t. $|\text{arcs}(G)| \leq H$ do
3: $E^* \leftarrow E^* \cup \text{Settlement-BB}$ on input $G$ \{Algorithm 1\}
4: for all $G \in \text{CC}$ s.t. $|\text{arcs}(G)| > H$ do
5: $E^* \leftarrow E^* \cup \text{Settlement-BEAM}$ on input $\langle G, K \rangle$ \{Algorithm 4\}

**Implementation details:**

- Extract the $(1, 1)$-$D$-core of the input S-multigraph beforehand
- Tree-like search space of Settlement-BB:
  - sort arcs by non-decreasing amount
  - DFS vs. BFS: no evident difference
Introduction: motivation, challenges, contributions

Service overview

Problem definition

Algorithms

- An exact algorithm
- A more efficient algorithm
- A hybrid algorithm

Experiments
Experimental evaluation: settings

- Random sample of a **real dataset** provided by UniCredit, a noteworthy European banking company
  - 5,413,375 receivables; 369,479 (anonymized) customers; 1 year in 2015-16

- Customers’ attributes set based on statistics computed on a training prefix of 3 months of data

- \( fl(u) = 0 \), for each customer \( u \)

- 6 simulation settings:
  - \( cap < \infty \) vs. \( cap = \infty \)
  - “worst”, “normal”, “best” scenarios (defined by invoice lifetime and \( cap \))

- \( L = 15 \) (all algorithms based on cycle enumeration), \( H = 20 \) (Settlement-\textbf{BEAM}), and \( K = 1,000 \) (Settlement-\textbf{BEAM} and Settlement-\textbf{HYBRID})
### Experimental evaluation: general performance

#### Settlement-bb-lb

<table>
<thead>
<tr>
<th>CAP</th>
<th>Scenario</th>
<th>Period</th>
<th>Amount</th>
<th>Time (s)</th>
<th>Receivables</th>
<th>Clients</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\infty$</td>
<td>normal</td>
<td>20150701-0930</td>
<td>553 364 544</td>
<td>64.58</td>
<td>7131</td>
<td>2836</td>
</tr>
<tr>
<td>$\infty$</td>
<td>normal</td>
<td>20151001-1231</td>
<td>643 722 123</td>
<td>6.67</td>
<td>6742</td>
<td>2736</td>
</tr>
<tr>
<td>$\infty$</td>
<td>normal</td>
<td>20160101-0331</td>
<td>693 852 990</td>
<td>29.03</td>
<td>7999</td>
<td>3034</td>
</tr>
<tr>
<td>$\infty$</td>
<td>normal</td>
<td>20160401-0630</td>
<td>751 368 135</td>
<td>30.81</td>
<td>8289</td>
<td>3189</td>
</tr>
</tbody>
</table>

#### Settlement-beam

<table>
<thead>
<tr>
<th>CAP</th>
<th>Scenario</th>
<th>Period</th>
<th>Amount</th>
<th>Time (s)</th>
<th>Receivables</th>
<th>Clients</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\infty$</td>
<td>normal</td>
<td>20150701-0930</td>
<td>660 907 304</td>
<td>1168.95</td>
<td>15 323</td>
<td>3761</td>
</tr>
<tr>
<td>$\infty$</td>
<td>normal</td>
<td>20151001-1231</td>
<td>663 873 349</td>
<td>618.98</td>
<td>14 570</td>
<td>3507</td>
</tr>
<tr>
<td>$\infty$</td>
<td>normal</td>
<td>20160101-0331</td>
<td>743 945 529</td>
<td>1159.27</td>
<td>17 390</td>
<td>4143</td>
</tr>
<tr>
<td>$\infty$</td>
<td>normal</td>
<td>20160401-0630</td>
<td>855 932 063</td>
<td>757.85</td>
<td>17 666</td>
<td>4155</td>
</tr>
</tbody>
</table>

#### Settlement-hybrid

<table>
<thead>
<tr>
<th>CAP</th>
<th>Scenario</th>
<th>Period</th>
<th>Amount</th>
<th>%Gain vs. $S$-bb-lb</th>
<th>%Gain vs. $S$-beam</th>
<th>Time (s)</th>
<th>Receivables</th>
<th>Clients</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\infty$</td>
<td>normal</td>
<td>20150701-0930</td>
<td>779 733K</td>
<td>40.91</td>
<td>17.98</td>
<td>1006.17</td>
<td>17 082</td>
<td>4268</td>
</tr>
<tr>
<td>$\infty$</td>
<td>normal</td>
<td>20151001-1231</td>
<td>784 315K</td>
<td>21.84</td>
<td>18.14</td>
<td>690.14</td>
<td>16 761</td>
<td>4133</td>
</tr>
<tr>
<td>$\infty$</td>
<td>normal</td>
<td>20160101-0331</td>
<td>827 346K</td>
<td>19.24</td>
<td>11.21</td>
<td>1329.80</td>
<td>19 544</td>
<td>4701</td>
</tr>
<tr>
<td>$\infty$</td>
<td>normal</td>
<td>20160401-0630</td>
<td>987 866K</td>
<td>31.48</td>
<td>15.41</td>
<td>865.92</td>
<td>19 576</td>
<td>4718</td>
</tr>
</tbody>
</table>
Experimental evaluation: scalability

Table: Scalability of the proposed Settlement-HYBRID algorithm

<table>
<thead>
<tr>
<th>Days</th>
<th>Nodes</th>
<th>Arcs</th>
<th>Amount</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>15 983</td>
<td>14 466</td>
<td>185 959</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>41 088</td>
<td>43 244</td>
<td>873 317</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>68 183</td>
<td>85 454</td>
<td>3 471 960</td>
<td>17</td>
</tr>
<tr>
<td>30</td>
<td>106 167</td>
<td>183 570</td>
<td>16 151 068</td>
<td>65</td>
</tr>
<tr>
<td>60</td>
<td>143 989</td>
<td>377 635</td>
<td>38 063 145</td>
<td>3291</td>
</tr>
<tr>
<td>90</td>
<td>168 861</td>
<td>600 172</td>
<td>73 101 255</td>
<td>27 504</td>
</tr>
</tbody>
</table>
We introduce a novel, network-based approach to receivable financing.

We provide a principled formulation and solution of such a novel service.

We define and characterize a novel optimization problem on a network of receivables, and design both an exact algorithm and a more efficient algorithm.

Experiments on real receivable data show that our algorithms work well in practice.

We believe our work is a well-suited example of how a real-world problem from a specific application domain (i.e., finance) requires non-trivial algorithmic and theoretical effort to be effectively solved in practice.
Thanks!
The Min-Cost Flow problem

Problem (Min-Cost Flow)

Given a simple directed graph $G = (V, E)$, a cost function $c : E \rightarrow \mathbb{R}$, lower-bound and upper-bound functions $\lambda : E \rightarrow \mathbb{R}$, $\mu : E \rightarrow \mathbb{R}$, and a supply/demand function $b : V \rightarrow \mathbb{R}$, find a flow $f : E \rightarrow \mathbb{R}$ so as to

Minimize $\sum_{e \in E} c(e)f(e)$

subject to $\lambda(e) \leq f(e) \leq \mu(e)$, $\forall e \in E$

$\sum_{u : (v, u) \in E} f(v, u) - \sum_{u : (u, v) \in E} f(u, v) = b(u)$, $\forall u \in V$
The **Set Union Knapsack** problem

Problem (**Set Union Knapsack**)

Let $U = \{x_1, \ldots, x_h\}$ be a universe of elements, $S = \{S_1, \ldots, S_k\}$ be a set of items, where $S_i \subseteq U$, $\forall i \in [1..k]$, $p : S \rightarrow \mathbb{R}$ be a profit function for items in $S$, and $q : U \rightarrow \mathbb{R}$ be a cost function for elements in $U$. For any $\hat{S} \subseteq S$ define also: $U(\hat{S}) = \bigcup_{S \in \hat{S}} S$, $P(\hat{S}) = \sum_{S \in \hat{S}} p(S)$, and $Q(\hat{S}) = \sum_{x \in U(\hat{S})} q(x)$. Given a real number $B \in \mathbb{R}$, **Set Union Knapsack** finds $S^* = \arg \max_{\hat{S} \subseteq S} P(\hat{S})$ s.t. $Q(\hat{S}) \leq B$. 

I. Bordino, F. Gullo

Network-based Receivable Financing
The **Multidimensional Set Union Knapsack** problem

**Problem (Multidimensional Set Union Knapsack)**

Given $U$, $S$, $p$ as in Set Union Knapsack, a $d$-dimensional cost function $q : U \to \mathbb{R}^d$, and a $d$-dimensional vector $B \in \mathbb{R}^d$, find $S^* = \arg \max_{\hat{S} \subseteq S} P(\hat{S}) \text{ s.t. } Q(\hat{S}) \leq B$, where

$$Q(\hat{S}) = \sum_{x \in U(\hat{S})} q(x).$$