Parameterized Approximability of Influence in Social Networks

Cristina Bazgan\textsuperscript{1,3}  Morgan Chopin\textsuperscript{1}  André Nichterlein\textsuperscript{2}  Florian Sikora\textsuperscript{1}

\textsuperscript{1}LAMSADE, Université Paris Dauphine, CNRS – France
\textsuperscript{2}TU Berlin – Germany
\textsuperscript{3}IUF

florian.sikora@dauphine.fr
Parameterized Approximability of Influence in Social Networks

Cristina Bazgan\textsuperscript{1,3} Morgan Chopin\textsuperscript{1} André Nichterlein\textsuperscript{2} Florian Sikora\textsuperscript{1}

\textsuperscript{1}LAMSADE, Université Paris Dauphine, CNRS – France
\textsuperscript{2}TU Berlin – Germany
\textsuperscript{3}IUF

florian.sikora@dauphine.fr
Parameterized Approximability of Influence in Social Networks

Cristina Bazgan\textsuperscript{1,3} \hspace{1cm} Morgan Chopin\textsuperscript{1} \hspace{1cm} André Nichterlein\textsuperscript{2} \hspace{1cm} Florian Sikora\textsuperscript{1}

\textsuperscript{1}LAMSADE, Université Paris Dauphine, CNRS – France
\textsuperscript{2}TU Berlin – Germany
\textsuperscript{3}IUF

florian.sikora@dauphine.fr
Parameterized Approximability of Influence in Social Networks

Cristina Bazgan¹,³ Morgan Chopin¹ André Nichterlein²
Florian Sikora¹

¹LAMSADÉ, Université Paris Dauphine, CNRS – France
²TU Berlin – Germany
³IUF

florian.sikora@dauphine.fr
Outline

Introduction

Hardness

FPT Approximation
Outline

Introduction

Hardness

FPT Approximation
An Example from Viral Marketing

Angela and Xi own an iPhone

Barack: “If three of my friends have an iPhone, I buy an iPhone too”

Nicolas: “If two of my friends have an iPhone, I buy an iPhone too”

Apple sold two iPhones without any advertisement!
An Example from Viral Marketing

- Angela and Xi own an iPhone
An Example from Viral Marketing

- Angela and Xi own an iPhone
- Barack: “If three of my friends have an iPhone, I buy an iPhone too”
**An Example from Viral Marketing**

- Angela and Xi own an iPhone
- Barack: “If three of my friends have an iPhone, I buy an iPhone too”
- Nicolas: “If two of my friends have an iPhone, I buy an iPhone too”
An Example from Viral Marketing

- Angela and Xi own an iPhone
- Barack: “If three of my friends have an iPhone, I buy an iPhone too”
- Nicolas: “If two of my friends have an iPhone, I buy an iPhone too”
**An Example from Viral Marketing**

- Angela and Xi own an iPhone
- Barack: “If three of my friends have an iPhone, I buy an iPhone too”
- Nicolas: “If two of my friends have an iPhone, I buy an iPhone too”
- Apple sold two iPhones without any advertisement!
An Example from Viral Marketing

- The first customers (target set) had an iPhone (Apple gave bonuses, free phones...).
- Goal: get the fewest customers with advertisement in order to attract the maximum number of customers at the end.
- What is the target set of customers to attract?
An Example from Viral Marketing

- The first customers (target set) had an iPhone (Apple gave bonuses, free phones...).
- Goal: get the fewest customers with advertisement in order to attract the maximum number of customers at the end.
- What is the target set of customers to attract?

Other applications:
- Spreading of information/influence in social networks via word-of-mouth recommendations.
- Diseases in populations.
- Faults in distributed computing.
- ...
Problem Definition

- **Diffusion** (threshold model):
  - A vertex of the graph is *activated* if it is in the *target set* or if at least $thr(v)$ of its *neighbors* are activated.

- **Optimization** problem [*Kempe et al. 2003*]:

""
Problem Definition

- **Diffusion** (threshold model):
  - A vertex of the graph is **activated** if it is in the **target set** or if at least $\text{thr}(v)$ of its **neighbors** are activated.

- **Optimization problem** [Kempe et al. 2003]:

  **Max Influence:**
  - **Input:** A graph, a threshold for each vertex, an integer $k$.
  - **Output:** A subset of vertices of size at most $k$ s.t. the number of activated vertices is maximum.
**Problem Definition**

- **Diffusion** (threshold model):
  - A vertex of the graph is *activated* if it is in the **target set** or if at least $thr(v)$ of its **neighbors** are activated.

- **Optimization** problem [Kempe et al. 2003]:

**Decision Influence:**

- **Input:** A graph, a threshold for each vertex, an integer $k$, an integer $l$.
- **Output:** A subset of vertices of size at most $k$ s.t. the number of activated vertices is **at least** $l$. 


Activation Process - Example

Numbers in vertex = threshold of the vertex
$k = 3$, maximize the number of activated vertices.
Activation Process - Example

Numbers in vertex = threshold of the vertex
$k = 3$, maximize the number of activated vertices.
Squares = target set
Activation Process - Example

Numbers in vertex = threshold of the vertex

\( k = 3 \), maximize the number of activated vertices.

Squares = target set
Activation Process - Example

Numbers in vertex = threshold of the vertex
$k = 3$, maximize the number of activated vertices.
Squares = target set
Activation Process - Example

Numbers in vertex = threshold of the vertex
$k = 3$, maximize the number of activated vertices.
Squares = target set
Activation Process - Example

Numbers in vertex = threshold of the vertex

$k = 3$, maximize the number of activated vertices.

Squares = target set
Thresholds

Different types of thresholds:

- General.
- Constant.
- Majority ($\text{thr}(v) = \lceil \text{degree}(v)/2 \rceil$).
- Unanimity ($\text{thr}(v) = \text{degree}(v)$).
Cardinality Constraint Problems

- Our problem is part of a larger class of problems formalized in [Cai 2008].
- Find a solution of **cardinality** $k$ (given in the input) s.t. an objective is **maximized** (or minimized).
- Examples:
  - **Max Vertex Cover**: Find $k$ vertices s.t. the number covered edges is maximum.
    - Classical **Vertex Cover** is FPT.
    - Decision version of **Max Vertex Cover** is W[1]-hard.
  - **Max Dominating Set**.
  - Same problems with minimization.
  - ...
(FPT)-Approximation – Better ratio

- Can achieve better ratios if we remove the polynomial-time constraint [Cai et al. 2006, Chen et al. 2006, Downey et al. 2006].
(FPT)-Approximation – Better ratio

- Can achieve better ratios if we remove the polynomial-time constraint [Cai et al. 2006, Chen et al. 2006, Downey et al. 2006].
- Problem is $f(n)$-approximable in fpt-time with respect to a parameter $k$ (could be anything...) if:
  - Algorithm achieve a $f(n)$-approximation.
  - With running time $g(k) \cdot n^c$. 

Pertinent for cardinality constraint problems!

Time parameterized by $k$.

Example for Max Vertex Cover:

No polynomial-time approximation scheme, W[1]-hard.

Admits a fpt-time approximation scheme [Marx 2008].
(FPT)-Approximation – Better ratio

- Can achieve better ratios if we remove the polynomial-time constraint [Cai et al. 2006, Chen et al. 2006, Downey et al. 2006].

- Problem is $f(n)$-approximable in fpt-time with respect to a parameter $k$ (could be anything...) if:
  - Algorithm achieve a $f(n)$-approximation.
  - With running time $g(k) \cdot n^c$.

- Pertinent for cardinality constraint problems!
  - Time parameterized by $k$.
  - Minimize/Maximize the objective.

- Example for Max Vertex Cover:
  - Admits a fpt-time approximation scheme [Marx 2008].
Known results

- (Of course), NP-hard, even in bipartite graphs and thresholds=2 [Chen 2009].
- **Hard to approximate** within $O(2^{\log^{1-\epsilon} n})$, even if thresholds are bounded by 2 and the graph is bipartite [Chen 2009].
- **W[2]-hard** for parameter solution size, even on majority and bounded thresholds [Nichterlein et al. 2012].
Known results

- (Of course), NP-hard, even in bipartite graphs and thresholds=2 \[\text{Chen 2009}\].

- **Hard to approximate** within \(O(2^{\log^{1-\epsilon} n})\), even if thresholds are bounded by 2 and the graph is bipartite \[\text{Chen 2009}\].

- **W[2]-hard** for parameter solution size, even on majority and bounded thresholds \[\text{Nichterlein et al. 2012}\].

- Our problem is hard to approximate and W[2]-hard:
  - Can we have better approximation ratio if we allow fpt-time?
Outline

Introduction

Hardness

FPT Approximation
Reduction

**Dominating Set**

- **Input:** A graph, an integer $k$.
- **Output:** A subset of the vertices of size at most $k$ s.t. each vertex of the graph has at least a neighbor in the solution.
Reduction

Dominating Set

- **Input:** A graph, an integer $k$.
- **Output:** A subset of the vertices of size at most $k$ s.t. each vertex of the graph has at least a neighbor in the solution.
Reduction

**Dominating Set**

- **Input:** A graph, an integer $k$.
- **Output:** A subset of the vertices of size at most $k$ s.t. each vertex of the graph has at least a neighbor in the solution.
Reduction

- From an instance of **Dominating Set** with parameter $k$.

![Graph](image.png)
Reduction

- From an instance of \textbf{Dominating Set} with parameter $k$.
- Build an instance of our problem.
  - Two copies of the vertex set.

\begin{itemize}
  \item $G = (V, E)$
  \item $G' = (V', E')$
  \item $V = \{v_1, v_2, v_3, v_4, v_5\}$
  \item $V' = \{v_1^t, v_2^t, v_3^t, v_4^t, v_5^t, v_1^b, v_2^b, v_3^b, v_4^b, v_5^b\}$
  \item Threshold = 1 for bottom vertices, degree($v$) for top ones.
\end{itemize}
Reduction

- From an instance of **Dominating Set** with parameter $k$.
- Build an instance of our problem.
  - Two copies of the vertex set with an edge in between.
Reduction

- From an instance of **Dominating Set** with parameter $k$.
- Build an instance of our problem.
  - Two copies of the vertex set with an edge in between.
  - For each edge $\{u, v\}$, add the edges $\{u^t, v^b\}$, $\{u^b, v^t\}$.
Reduction

- From an instance of **Dominating Set** with parameter \( k \).
- Build an instance of our problem.
  - Two copies of the vertex set with an edge in between.
  - For each edge \( \{u, v\} \), add the edges \( \{u^t, v^b\}, \{u^b, v^t\} \).

\[
\begin{align*}
G & \quad G' \\
\begin{array}{c}
\bullet v_1 \\
\bullet v_2 \\
\bullet v_3 \\
\bullet v_4 \\
\bullet v_5 \\
\end{array} & \quad \begin{array}{c}
\bullet v_{1^t} \\
\bullet v_2^t \\
\bullet v_3^t \\
\bullet v_4^t \\
\bullet v_5^t \\
\bullet v_{1^b} \\
\bullet v_2^b \\
\bullet v_3^b \\
\bullet v_4^b \\
\bullet v_5^b \\
\end{array}
\end{align*}
\]
**Reduction**

- From an instance of **DOMINATING SET** with parameter $k$.
- Build an instance of our problem.
  - Two copies of the vertex set with an edge in between.
  - For each edge $\{u, v\}$, add the edges $\{u^t, v^b\}$, $\{u^b, v^t\}$.
Reduction

- From an instance of **Dominating Set** with parameter $k$.
- Build an instance of our problem.
  - Two copies of the vertex set with an edge in between.
  - For each edge $\{u, v\}$, add the edges $\{u^t, v^b\}$, $\{u^b, v^t\}$.
  - Threshold = 1 for bottom vertices, degree($v$) for top ones.
Reduction

- From an instance of \texttt{Dominating Set} with parameter $k$.
- Build an instance of our problem.
  - Two copies of the vertex set with an edge in between.
  - For each edge $\{u, v\}$, add the edges $\{u^t, v^b\}$, $\{u^b, v^t\}$.
  - Threshold $= 1$ for bottom vertices, degree($v$) for top ones.

\begin{itemize}
  \item Dominating set of size $k \Rightarrow$ Target set of size $k$ activating all the vertices.
\end{itemize}
Reduction

- From an instance of **Dominating Set** with parameter $k$.
- Build an instance of our problem.
  - Two copies of the vertex set with an edge in between.
  - For each edge $\{u, v\}$, add the edges $\{u^t, v^b\}$, $\{u^b, v^t\}$.
  - Threshold $= 1$ for bottom vertices, $\text{degree}(v)$ for top ones.

- Dominating set of size $k \Rightarrow$ Target set of size $k$ activating all the vertices.
Reduction

- From an instance of **Dominating Set** with parameter $k$.
- Build an instance of our problem.
  - Two copies of the vertex set with an edge in between.
  - For each edge $\{u, v\}$, add the edges $\{u^t, v^b\}$, $\{u^b, v^t\}$.
  - Threshold $= 1$ for bottom vertices, degree($v$) for top ones.

![Diagram of graphs $G$ and $G'$]

- Dominating set of size $k \Rightarrow$ Target set of size $k$ activating all the vertices.
Reduction

- From an instance of **Dominating Set** with parameter $k$.
- Build an instance of our problem.
  - Two copies of the vertex set with an edge in between.
  - For each edge $\{u, v\}$, add the edges $\{u^t, v^b\}, \{u^b, v^t\}$.
  - Threshold $= 1$ for bottom vertices, degree$(v)$ for top ones.

\[
\text{G} \\
\begin{array}{c}
    v_1 \\
    v_2 \\
    v_3 \\
    v_4 \\
    v_5
\end{array}
\]

\[
\text{G'} \\
\begin{array}{ccc}
    v_1^t & v_2^t & v_3^t \\
    v_1^b & v_2^b & v_3^b \\
    v_4^t & v_5^t & v_4^b & v_5^b
\end{array}
\]

- Dominating set of size $k \Rightarrow$ Target set of size $k$ activating all the vertices.
Reduction

- From an instance of Dominating Set with parameter $k$.
- Build an instance of our problem.
  - Two copies of the vertex set with an edge in between.
  - For each edge $\{u, v\}$, add the edges $\{u^t, v^b\}$, $\{u^b, v^t\}$.
  - Threshold $= 1$ for bottom vertices, degree($v$) for top ones.

![Graph $G$ and $G'$ with vertex sets and edge configurations.]

- Dominating set of size $k$ $\Rightarrow$ Target set of size $k$ activating all the vertices.
- Target set of size $k$ activating all the vertices $\Rightarrow$ Dominating set of size $k$. 
Reduction

- From an instance of Dominating Set with parameter $k$.
- Build an instance of our problem.
  - Two copies of the vertex set with an edge in between.
  - For each edge $\{u, v\}$, add the edges $\{u^t, v^b\}$, $\{u^b, v^t\}$.
  - Threshold $= 1$ for bottom vertices, $\text{degree}(v)$ for top ones.

- Dominating set of size $k \Rightarrow$ Target set of size $k$ activating all the vertices.
- Target set of size $k$ activating all the vertices $\Rightarrow$ Dominating set of size $k$. 
Reduction

- From an instance of **Dominating Set** with parameter $k$.
- Build an instance of our problem.
  - Two copies of the vertex set with an edge in between.
  - For each edge $\{u, v\}$, add the edges $\{u^t, v^b\}$, $\{u^b, v^t\}$.
  - Threshold = 1 for bottom vertices, degree($v$) for top ones.

- Dominating set of size $k$ $\Rightarrow$ Target set of size $k$ activating all the vertices.
- Target set of size $k$ activating all the vertices $\Rightarrow$ Dominating set of size $k$. 
Parameterized Intractibility

- With **additional gadgets**, we can prove that our problem cannot be approximated within $n^{1-\epsilon}$ in **fpt-time** unless $\text{FPT} = \text{W}[2]$.
- Even if:
  - The graph is bipartite.
  - Majority thresholds.
  - Thresholds are at most 2.
  - All the activated vertices (including the target set) are counted.
Outline

Introduction

Hardness

FPT Approximation
Unanimity Thresholds

- All the neighbors of a vertex must be activated.
- Only one round!
Unanimity Thresholds

- All the neighbors of a vertex must be activated.
  - Only one round!
Unanimity Thresholds

- All the neighbors of a vertex must be activated.
  - Only one round!
Unanimity Thresholds

- All the neighbors of a vertex must be activated.
- Only one round!
Unanimity Thresholds

- We prove that the problem is $W[1]$-hard for combined parameter $(k, l)$ even for bipartite graphs.
- **Cannot** be approximated within $n^{1-\epsilon}$ in *polynomial time*...
We prove that the problem is $W[1]$-hard for combined parameter $(k, l)$ even for bipartite graphs.

- **Cannot** be approximated within $n^{1-\epsilon}$ in polynomial time...
- But it is $2^k$-approximable in polynomial time!
We prove that the problem is \( \text{W}[1] \)-hard for combined parameter \((k, l)\) even for bipartite graphs.

\textbf{Cannot} be approximated within \( n^{1-\epsilon} \) in \textit{polynomial time}...

But it is \( 2^k \)-\textit{approximable in polynomial time}!

Therefore, it is \( r(n) \)-\textit{approximable in fpt-time}, for any strictly increasing function \( r \).
Approximation in fpt-time within arbitrarily small ratios

- We know that our problem is $2^k$-approximable in polynomial-time.
- We want to prove that it is also $\log_2(n)$-approximable in fpt-time.
Approximation in fpt-time within arbitrarily small ratios

- We know that our problem is $2^k$-approximable in polynomial-time.
- We want to prove that it is also $\log_2(n)$-approximable in fpt-time.
- Distinguish two cases for our problem.
Approximation in fpt-time within arbitrarily small ratios

- We know that our problem is $2^k$-approximable in polynomial-time.
- We want to prove that it is also $\log_2(n)$-approximable in fpt-time.
- Distinguish two cases for our problem.
  - $k < \log_2 \log_2(n)$:
    - The $2^k$-approximation becomes a $2^{\log_2 \log_2 n} = \log_2 n$-approximation (in polynomial time).
Approximation in fpt-time within arbitrarily small ratios

- We know that our problem is \(2^k\)-approximable in polynomial-time.
- We want to prove that it is also \(\log_2(n)\)-approximable in fpt-time.
- Distinguish two cases for our problem.
  - \(k < \log_2 \log_2(n)\):
    - The \(2^k\)-approximation becomes a \(2^{\log_2 \log_2 n} = \log_2 n\)-approximation (in polynomial time).
  - \(k > \log_2 \log_2(n) \Rightarrow n < 2^{2^k}\).
    - Apply any brute-force algorithm testing all subsets of size \(k\) for the solution and take the one making the better solution. **Exact algorithm in fpt-time.**
Approximation in fpt-time within arbitrarily small ratios

- We know that our problem is $2^k$-approximable in polynomial-time.
- We want to prove that it is also $\log_2(n)$-approximable in fpt-time.
- Distinguish two cases for our problem.
  - $k < \log_2 \log_2(n)$:
    - The $2^k$-approximation becomes a $2^{\log_2 \log_2 n} = \log_2 n$-approximation (in polynomial time).
  - $k > \log_2 \log_2(n) \Rightarrow n < 2^{2^k}$.
    - Apply any brute-force algorithm testing all subsets of size $k$ for the solution and take the one making the better solution.
      **Exact algorithm in fpt-time.**
- All together: approximation algorithm in fpt-time ($\log_2(n)$-approximation in time $O^*(2^{k2^k})$).
Approximation in fpt-time within arbitrarily small ratios

- We know that our problem is $2^k$-approximable in polynomial-time.
- We want to prove that it is also $\log_2(n)$-approximable in fpt-time.
- Distinguish two cases for our problem.
  - $k < \log_2 \log_2(n)$:
    - The $2^k$-approximation becomes a $2^\log_2 \log_2 n = \log_2 n$-approximation (in polynomial time).
  - $k > \log_2 \log_2(n) \Rightarrow n < 2^{2^k}$.
    - Apply any brute-force algorithm testing all subsets of size $k$ for the solution and take the one making the better solution.
      Exact algorithm in fpt-time.

- All together: approximation algorithm in fpt-time ($\log_2(n)$-approximation in time $O^*(2^{k2^k})$).
- Generalization: replace $\log_2(n)$ by any strictly increasing function of $n$.
  - A worse running time implies a better ratio.
Approximation in fpt-time within arbitrarily small ratios

- Apply to all cardinality constraint problems.
Approximation in fpt-time within arbitrarily small ratios

- Apply to all cardinality constraint problems.
- We can make a E-reduction from Densest $k$-Subgraph to our problem with unanimity thresholds.
- $\Rightarrow$ Densest $k$-Subgraph is also $r(n)$-approximable in fpt-time, for any increasing function $r$. 
Conclusion

- Problem hard to approximate, even in fpt-time.
- If the thresholds are unanimity, the problem is a bit easier.
- In the paper, more positive results (approximation, fpt) if we focus on bounded degree graphs.
MERCI
Fixed-Parameter Tractability

“Measuring complexity only in terms of the input size means ignoring any structural information about the instances”

J. Flum et M. Grohe

“Question : When will the input of a problem coming from “real life” have no more structure than its size?
Answer : Never!”

R. Downey et M. Fellows
Fixed-Parameter Tractability

“Measuring complexity only in terms of the input size means ignoring any structural information about the instances”
J. Flum et M. Grohe

“Question : When will the input of a problem coming from “real life” have no more structure than its size?
Answer : Never!”
R. Downey et M. Fellows

“The fundamental idea is to restrict the combinatorial explosion, seemingly unavoidable, that causes the exponential growth in the running time of certain problem-specific parameters…”
R. Niedermeier
Fixed-Parameter Tractability

- Examples:
  - Solution size $k$ in a $n$-vertices graph.
  - $n$ voters for $k$ candidates.
  - Requests of size $k$ in a $n$-sized database.
  - ...
Fixed-Parameter Tractability

► Examples:
  ▶ Solution size $k$ in a $n$-vertices graph.
  ▶ $n$ voters for $k$ candidates.
  ▶ Requests of size $k$ in a $n$-sized database.
  ▶ ...

► A problem is in the class FPT if any instance $(I, k)$ can be solved exactly in $f(k) \cdot |I|^c$. 
Fixed-Parameter Tractability

- Examples:
  - Solution size $k$ in a $n$-vertices graph.
  - $n$ voters for $k$ candidates.
  - Requests of size $k$ in a $n$-sized database.
  - ...

- A problem is in the class FPT if any instance $(I, k)$ can be solved exactly in $f(k) \cdot |I|^c$.

- Complexity classes: $\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \ldots$
Approximation

- For an NP-hard optimisation problem, no polynomial-time algorithm is possible (unless P = NP)...
- ... if the solution must be exact!
Approximation

- For an NP-hard optimisation problem, no polynomial-time algorithm is possible (unless P = NP)...
- ... if the solution must be exact!
- Allow errors to obtain a polynomial-time algorithm.
- With a bound on the error.
Approximation

- For an NP-hard optimisation problem, no polynomial-time algorithm is possible (unless P = NP)...
- ... if the solution must be exact!
- Allow errors to obtain a polynomial-time algorithm.
- With a bound on the error.
- An algorithm is a $f(n)$-approximation if it runs in polynomial-time and if the cost of the returned solutions is bounded in the worst case by $r \times Opt$ (for minimization problems).

![Diagram showing a range of solutions from $Opt$ to $f(n) \times Opt$](image)
Unanimity Thresholds - A $2^k$ approximation in poly-time

- Find the largest set of “false-twins” with all vertices degree bounded by $k$.
- Make the neighbors of this set as the solution (the target set)
  - There is at most $k$ neighbors.
  - This will activate all the vertices in the false-twins set in the next round.
Find the largest set of “false-twins” with all vertices degree bounded by $k$.

Make the neighbors of this set as the solution (the target set)
  - There is at most $k$ neighbors.
  - This will activate all the vertices in the false-twins set in the next round.

There is at most $2^k$ different false-twins set.