Technische Universität Berlin

Institute of Software Engineering and
Theoretical Computer Science
Fachgebiet Algorithmik und Komplexitätstheorie

Fakultät IV
Franklinstrasse 28-29
10587 Berlin

Bachelor Thesis

Paths under Neighborhood Constraints — Algorithms and Complexity

Max-Jonathan Luckow

May 2, 2017

Supervised by
Prof. Dr. Rolf Niedermeier

Till Fluschnik

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den 02.05.2017

. . . . . .
Unterschrift

Zusammenfassung

Im Rahmen einer Modernisierung eines Netzwerks mit Hilfe einer neuen Technologie, wie zum Beispiel von Glasfaserkabeln, ist es meist wünschenswert sie gezielt dort einzusetzen, wo es die meisten Menschen erreicht. Wir suchen also nach einem Pfad, dessen Einwirkung auf die Umgebung besonders groß ist und untersuchen Variationen dieses Problems, in denen es einerseits Anforderungen an die Größe des Pfades und andererseits an die Umgebung des Pfades gibt. Erstaunlicherweise sind alle von uns beleuchteten Variationen NP-vollständig. Wir präzisieren dieses Ergebnis sowohl hinsichtlich der parametrisierten Komplexität als auch bezüglich einiger Graphklassen.

Abstract

Imagine a setting where a new technology such as fiber optic cables are supposed to modernize a network. However the supply is still limited and at first one should introduce this new technology to areas where one reaches the most people, in order to have the most effect. Hence we are looking for a path where the *exposure* to the surroundings is high. We study variations of this kind of problem where there is a demand in the size of a path but also in the exposure of it to the network. Surprisingly, it turns out that all of the variations we examined are NP-complete. We present further investigation in terms of their parametrized complexity as well as some results on specific graph classes.

Contents

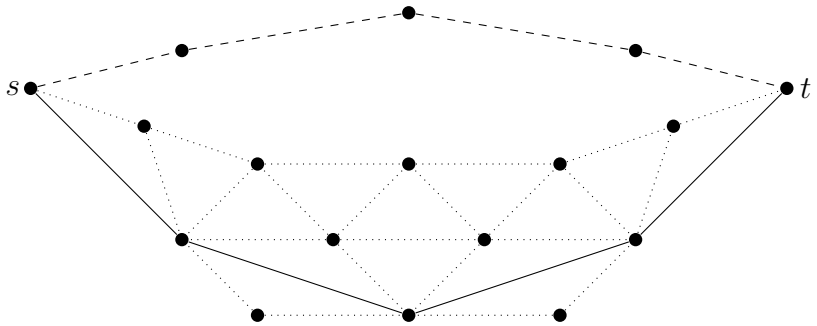

Figure 1: A graph with two terminal vertices s and t and two shortest paths with length 4. The neighborhood of the solidly drawn path consists of 10 vertices, whereas the neighborhood of the dashed path only consists of 4 vertices. The remaining dotted lines are the edges of the graph, which are not included in either path.

1 Introduction

Finding the shortest path between two vertices in a graph is one of the basic problems in graph theory. A polynomial-time algorithm to find such a path was already given by Dijkstra [7] in 1959. Consider a graph with two terminal vertices s and t given in Figure 1. There are two shortest paths with both the same length. But they differentiate each other by a property, which might be of interest in some applications. For instance those two terminals could be cities which shall be connected by a new high speed railroad line. However it is advisable to reduce the amount of noise or pollution to the surrounding settlements. In that case one tries to keep the costs low by keeping the size of the track relatively small and also avoid a high exposure to the surroundings. A lot of problems can be modeled using a graph and then solved with graph algorithms. Common uses of graphs can be found in the area of networks connecting computers or in the area of transportation. As seen in the Figure 1 the neighborhood, that is the adjacent vertices of the two possible paths are different. The dashed path is more *secluded* than the other one, which might be of interest. However the algorithm given by Dijkstra [7] does only take the length into account and does not prefer one solution over the other if they both share the same length.

We are going to study four new path problems. These are, in contrast to optimization, decision type problems of finding a path with certain properties. One can imagine looking for a path with length below or above a certain threshold, denoted Short or Long paths in the following work. The second property we look at is the neighborhood of the path. If the goal is to find a path with a neighborhood below a certain threshold it is called Secluded; if a greater neighborhood is desirable it is denoted Unsecluded. By combining these two possible issues one arrives at four problem definitions, which are going to be the

main topic of this thesis.

Input:	An undirected graph $G = (V, E)$, integers k, ℓ.
Question:	Is there a path $P = (V_p, E_p)$ in G such that
	SHORT SECLUDED PATH: $\lvert V_p\rvert \leq k, \lvert N_G(V_p)\rvert \leq \ell$?
	SHORT UNSECLUDED PATH: $\lvert V_p\rvert \leq k, \lvert N_G(V_p)\rvert \geq \ell$?
	LONG SECLUDED PATH: $\lvert V_p\rvert \geq k, \lvert N_G(V_p)\rvert \leq \ell$?
	LONG UNSECLUDED PATH: $\lvert V_p\rvert \geq k, \lvert N_G(V_p)\rvert \geq \ell$?

Related work. To examine common problems with a secluded variation was started by Chechik et al. [4]. Their motivation was sending sensitive information over a network thus trying to minimize the overall exposure to the graph, meaning the closed neighborhood. On the one hand they applied this framework on the PATH problem and also the STEINER TREE problem, which is the problem of finding a tree connecting a number of terminal vertices. SECLUDED PATH and SECLUDED STEINER TREE were proven to be NP-complete. They also showed a polynomial-time algorithm for SECLUDED PATH on graphs with bounded degree or graphs with bounded treewidth. Those problems were further analyzed by Fomin et al. [11] in terms of parametrized complexity. Among others they were able to show that the problems are fixed-parameter tractable if parametrized by the size of the exposure. Other problems under neighborhood constraints were considered by Fomin, Golovach, and Korhonen [10], which introduced two parameters, one for the size of the solution and one for, contrary to the work by Fomin et al. [11] the open neighborhood, not including the solution itself. They looked at problems where the open neighborhood had to be smaller than a given parameter. This framework, then baptized as SMALL SECLUDED was closer studied van Bevern et al. [2], by applying it to various other problems for example the s-t SEPARATOR for their complexity and parametrized complexity. They found that SECLUDED s-t SEPARATOR is polynomial-time solvable, however that SMALL SECLUDED s-t SEPARATOR is NP-hard. They also expanded the framework by including LARGE SECLUDED INDEPENDENT SET, where the solution size shall be big, while the open neighborhood has to be kept below a given parameter. They suggested that these keywords together with UNSECLUDED should be analyzed further for more common problems. Our work builds on that going back to the original PATH problem and analyzing it under this modified framework.

Our results. Surprisingly, all of our four main problems SHORT SECLUDED PATH, SHORT UNSECLUDED PATH, LONG SECLUDED PATH and LONG UNSECLUDED PATH as well as the s-t-variants given to each problem turn out to be NP-complete. Those results are proven in Section 3 with a more detailed explanation. In Section 4 those results are further analyzed in terms of their parametrized complexity. In general they appear to be hard for a practical point of view, however there is still some hope in the cases of the short variants. For

Path Problem	Compl.	Parametrized Complexity		
		ℓ	k	$k+\ell$
Short Secluded	NP-c.	NP wcp.	XP, W[1]-hard	?
Short Unsecluded	NP-c.	?	XP, W[2]-hard	FPT
Long Secluded	NP-c.	NP wcp.	NP wcp.	NP wcp.
Long Unsecluded	NP-c.	NP wcp.	W[2]-hard	?
Short Secluded s-t	NP-c.	?	W[1]-hard	?
Short Unsecluded s-t	NP-c.	?	W[2]-hard	?
Long Secluded s-t	NP-c.	?	NP wcp.	?
Long Unsecluded s-t	NP-c.	?	W[2]-hard	?

Table 1: Overview of our results - NP-c. stands for "NP-complete" and NP wcp. stands for "NP-complete with constant parameter".

Short Unsecluded Path there even is a FPT algorithm with respect to k and l. We refer to Table 1 for an overview of our results. Our problems appear to be closely related to Hamiltonian Path and thus we were able to prove that the problems are NP-complete on the same graph classes. We assume that they are polynomial-time solvable on the same graph classes as well. We were able to give polynomial-time algorithms on cographs and threshold graphs. We refer to Figure 2 for an overview concerning the classification on graph classes.

2 Basic Definitions and Preliminaries

We use basic definitions and notations given by Diestel [6]. We only consider finite, undirected graphs that do not contain loops or multiple edges between two vertices: A *graph* $G = (V, E)$ is a pair of a set V of *vertices* and a set E of *edges*. We denote $G(V) = V$ and $G(E) = E$. An *edge* $e \in E$ is a set of different vertices $\{x, v\}$ or xv for short where $x, v \in V$. We denote the number of vertices as $n := |V|$ and the number of edges as $m := |E|$.

Neighborhoods. The set of *neighbors* of a vertex v is denoted by $N_G(v)$, where $x \in N_G(v)$ if and only if there is an edge $xv \in E$. For a set $X \subseteq V$ of vertices the *open neighborhood* is $N_G(X) = \bigcup_{x \in X} N_G(x) \setminus X$. The *closed neighborhood* is $N_G[X] = N_G(X) \cup X$. The *degree* of a vertex v is the number of its neighbors $d_G(v) = |N_G(v)|$. A vertex v is called *isolated* when it holds $d_G(v) = 0$. A vertex v is called *dominating* when it holds $d_G(v) = n - 1$. The *minimum degree* of G is $\delta(G) = \min\{d(v) \mid v \in V\}$ and the *maximum degree* of G is $\Delta(G) = \max\{d(v) \mid v \in V\}$.

Graph properties. A v_1-v_k *path* $P = (V_P, E_P)$ is a non-empty graph with $V_P = \{v_1, v_2, \ldots, v_k\}$ and $E_P = \{v_1v_2, v_2v_3, v_{k-1}v_k\}$, where each vertex $v \in V$ is

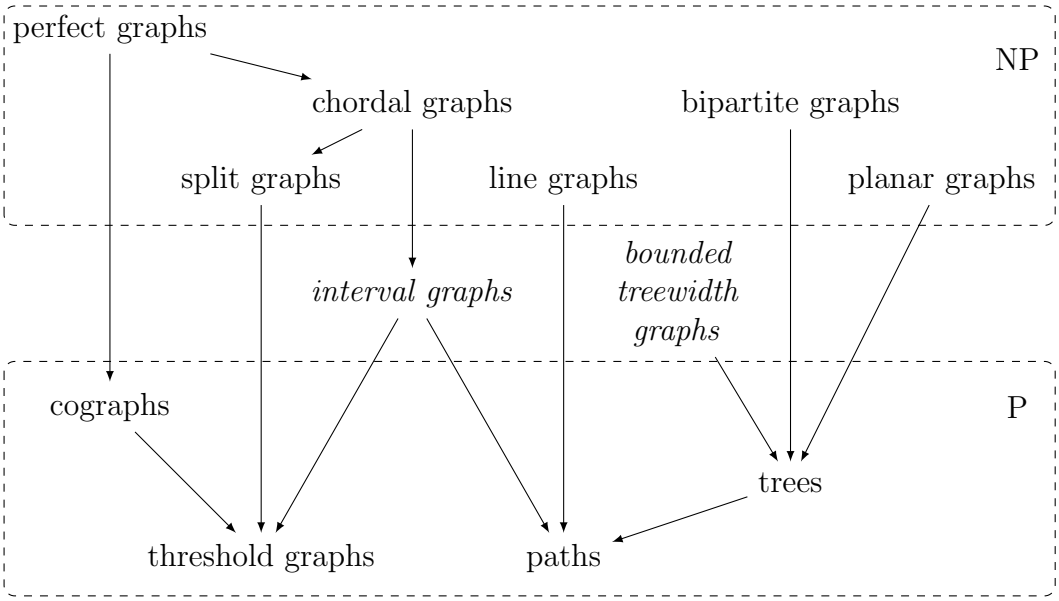

Figure 2: Overview of our results on graph classes for most of our variants. If there is an arrow from $\mathcal{C}_i$ to $\mathcal{C}_j$ then $\mathcal{C}_j \subseteq \mathcal{C}_i$. The problems on graph classes written in *italics* are assumed to be contained in P but not yet known.

distinct. The number $|E_P|$ of edges of a path is its *length*. A path with length $|E_P| = 0$ contains only one vertex. A non-empty graph G is called *connected*, if for any two vertices $v, w \in V$ there is a v-w path. A *component* of G is a maximal connected subgraph. A graph $G = (V, E)$ is *isomorphic* to another graph $G' = (V', E')$ if there exists a bijection $\varphi : V \to V'$ with $xy \in E \Leftrightarrow \varphi(x)\varphi(y) \in E'$ for all $x, y \in V$. An *independent set* $I \subseteq G$ is a set of vertices such that no two vertices of I are adjacent to each other. Whereas a *clique* $C \subseteq G$ is a set of vertices such that any two vertices of C are adjacent to each other.

Graph classes. A graph G is a *tree* if G is connected and acyclic. The *leafs* of a tree are those vertices of degree 1. A *tree-decomposition* of a Graph G is a tree with *bags* X as vertices, where $X \subseteq V(G)$ with the following three properties. The union of all bags have to be $V(G)$. For each edge $e = uv \in E(G)$ there has to be a bag X with $u, v \in X$. Finally each subset $S \subset V(G)$, the bags which include any vertex of S have to form a subtree of the tree-decomposition of G. The *width of a tree-decomposition* is $|X_b| - 1$, where X_b is a bag with the most vertices of the tree-decomposition. The *treewidth* of a graph G is the minimum width over all possible tree-decompositions of G [3]. A graph G is *bipartite* if $V(G)$ can be partitioned into two sets U and W such that each edge connects U and W [25]. A graph G is a *split graph* when $V(G)$ can be partitioned into an independent set and a clique [1]. A *line graph* $L(G)$ can be constructed from G by creating a vertex for each edge and two vertices of $L(G)$ have an edge

between them when the corresponding edges are adjacent [12]. The two edges $e = uv$ and $e' = uw$ are adjacent to each other, because they share the vertex u. A graph G is called *planar* if it has an embedding in the plane $\mathbb{R}^2$, such that no two edges cross each other [25]. A graph G is an interval graph if the vertices of G can be mapped on intervals of the real line such that there is an edge between u and v when the corresponding intervals of u and v intersect each other [21].

Parametrized Complexity. The field of parametrized complexity was mainly initiated by Downey and Fellows [8]. An overview over the field can be further read in the books of Flum and Grohe [9] and Niedermeier [24]. The parametrized complexity is the approach of studying a given problem not only based on the size of the input n, but also on another parameter k. A problem is a *parametrized problem* parametrized by k if it has a parameter k with its value being fixed. A parametrized problem is *fixed-parameter tractable*, or contained in FPT, if it can be solved in $f(k) \cdot n^{\mathcal{O}(1)}$ time, whereas the complexity class XP contains problems which can be solved in $f(k) \cdot n^{g(k)}$ time with computable functions f and g. Clearly, FPT $\subseteq$ XP and it even has been shown that FPT $\subset$ XP [8]. In between FPT and XP lie a hierarchy of parametrized intractability classes called W[1] and W[2] and others. It is known that FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \cdots \subseteq$ XP [8]. However similarly to the $P \neq$ NP conjecture it is not known whether these subsets are strict. It is assumed though, that FPT $\neq$ W[1] and is thus unlikely that a W[1]-hard problem would have a running time of $f(k) \cdot n^{\mathcal{O}(1)}$. A *parametrized reduction* is from a problem A to a problem B is a function ϕ which can be computed in time $f(k) \cdot |x|^{\mathcal{O}(1)}$ where x is a yes-instance of A if any only if $\phi(x)$ is a yes-instance of B. To show W[1]-hardness one has to give a parametrized reduction to a problem which is known to be W[1]-hard.

3 NP-hardness

Contrarily to what one might imagine, it turns out that the four variants Short Secluded Path, Short Unsecluded Path, Long Secluded Path and Long Unsecluded Path are very similar to each other. One could have speculate that at least the short variants would be easier to solve from a complexity standpoint, since the problem of finding the shortest path between two vertices s and t is polynomial time solvable. It turns out that in terms of our path problems the secluded and unsecluded framework is enough to make those problems NP-complete. What jumps out though is, that we deem it necessary to change the graph of the reduction only for Short Unsecluded Path, which has the effect that not all of the results from Hamiltonian Path concerning the graph classes carries over to Short Unsecluded Path.

Theorem 3.1. *All four* Short Secluded Path, Short Unsecluded Path, Long Secluded Path, Long Unsecluded Path *are* NP*-complete.*

Proof. Given a solution $P = (V, E)$ of any of Short Secluded Path, Short Unsecluded Path, Long Secluded Path, Long Unsecluded Path, can be verified in polynomial time by checking the length and the neighborhood of the path against the parameters k and ℓ, meaning they are contained in NP. The NP-hardness results are shown in Proposition 3.2, Proposition 3.3, Proposition 3.4 and Proposition 3.5 for the individual problems. □

For the following proofs we will need two problems from which we can reduce in order to proof NP-hardness.

Hamiltonian Path	
Input:	An undirected graph $G = (V, E)$.
Question:	Is there a path in G containing each vertex in V exactly once ?

The Hamiltonian Path problem is known to be NP-complete [13]. Furthermore it stays NP-hard on split [23], bipartite [14] and line graphs [15] as well as on planar graphs with bounded degree $\Delta \geq 3$ [18]. Note that whenever G is not connected G will always be a trivial no-instance of Hamiltonian Path.

Long Path	
Input:	An undirected graph $G = (V, E)$, integer k.
Question:	Is there a path $P = (V_p, E_p)$ in G such that $\lvert V_p \rvert \geq k$?

The second NP-complete problem is Long Path which can be considered, in a way, as a more general case of the Hamiltonian Path, since if one imagines an instance $(G, k = n)$ of Long Path with $|V(G)| = n$ it resembles the Hamiltonian Path. Interestingly Long Path is contained in FPT when parametrized by the length of the path k [22].

Proposition 3.2. Short Secluded Path *is* NP*-hard on split, bipartite, line and planar graphs even if the parameter* $\ell = 0$.

Proof. We reduce from Hamiltonian Path, which is only defined on connected graphs. Let G be an instance of Hamiltonian Path with $|V(G)| = n$.
Construction. We construct an instance $(G', k = n, \ell = 0)$ of Short Secluded Path such that G is isomorphic to $G' = (V', E')$. We claim that G is a yes-instance of Hamiltonian Path if and only if (G', k, ℓ) is a yes-instance of Short Secluded Path.
"$\Rightarrow$". Assume there is a Hamiltonian path $P_h = (V_h, E_h)$ in the graph G. Then $N_G(V_h) = 0$ and $|V_h| = n$ due to the definition of Hamiltonian Path. As G' is isomorphic to G, P_h is also a solution to Short Secluded Path with $k = n$ and $\ell = 0$.
"$\Leftarrow$". Assume there is a path $P_s = (V_s, E_s)$ with $|V_s| \leq n$ and $|N_{G'}(V_s)| \leq 0$. Suppose $|V_s| < n$. Then there is a vertex $v \in V'$ with $v \notin V_s$. Since the graph G' is connected, there has to be a $w \in V_s$ such that a v-w path exists. Hence $N_{G'}(V_s) \geq 1$ and thus contradicting our assumption. The solution P_s forms a Hamiltonian path since $|V_s| = n$.

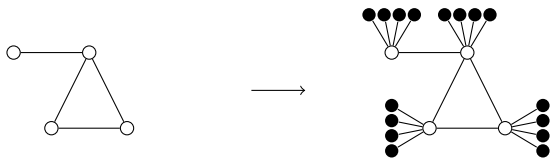

Figure 3: Construction of G' used in the proof of Proposition 3.3

Since G' is isomorphic to G the NP-hardness results from Hamiltonian Path concerning the graph classes carries over to Short Secluded Path. As we set $l = 0$ in the reduced instance, it follows that Short Secluded Path is NP-complete with a constant parameter $\ell = 0$. □

Proposition 3.3. Short Unsecluded Path *is* NP*-hard even on bipartite and planar graphs.*

Proof. We reduce from Hamiltonian Path. Let G be an instance of Hamiltonian Path with $|V(G)| = n$.
Construction. We construct an instance $(G', k = n, \ell = n^2)$ of Short Unsecluded Path where G' is obtained from G by attaching n vertices with no other neighbors to every vertex of G. We refer to Figure 3 for an Illustration. We claim that G is a yes-instance of Hamiltonian Path if and only if (G', k, ℓ) is a yes-instance of Short Unsecluded Path.
"$\Rightarrow$". Assume there is a Hamiltonian path in G. Let $P_h = (V_h, E_h)$ bet the corresponding path in G'. We show that P_h is also a short unsecluded path with respect to k and ℓ. Since $|V_h| = n$ and all of those vertices having n neighbors it follows that $N_{G'}(V_h) = n^2$.
"$\Leftarrow$". Assume there is a path $P_s = (V_s, E_s)$ with $|V_s| \leq n$ and $|N_{G'}(V_s)| \geq n^2$ in G'. Then the neighborhood of that path is at most $|N_{G'}(V_s)| \leq |V_s| \cdot n + n - |V_s|$. Since it holds that $|N_{G'}(V_s)| \geq n^2$ we can conclude that if $n > 1$ then $|V_s| \geq n$. Furthermore V_s has to contain every original vertex in order to gain the n^2 added isolated vertices in its neighborhood and cannot contain any additional ones (from the isolated vertices) since it would decrease $|N_{G'}(V_s)|$. Thus the corresponding path in G of P_s is a Hamiltonian path. The graph G' used in our reduction remains a bipartite and planar if G was as well, because we are only adding single vertices connected by one edge to another vertex. □

Proposition 3.4. Long Secluded Path *is* NP*-hard on split, bipartite, line and planar graphs even if* $\ell = 0$ *and* k *has any constant value.*

Remark. Since Long Secluded Path and Short Secluded Path are closely related the used reductions are similar to each other. The difference lies in the parameter k which had to be adjusted to accompany the different behavior of the path length.

Proof. We reduce from Hamiltonian Path, which is only defined on connected graphs. Let G be an instance of Hamiltonian Path with $|V(G)| = n$.

Construction. We construct an instance $(G', k = 0, \ell = 0)$ of Long Secluded Path such that G is isomorphic to $G' = (V', E')$. We claim that G is a yes-instance of Hamiltonian Path if and only if (G', k, ℓ) is a yes-instance of Long Secluded Path.
"$\Rightarrow$". Assume there is a Hamiltonian path $P_h = (V_h, E_h)$ in the graph G. Then $|V_h| = n > 0$ and $N_G(V_h) = \emptyset$ by definition of the Hamiltonian path, meaning that the same path is also a solution for Long Secluded Path with $k = 0$ and $\ell = 0$.
"$\Leftarrow$". Assume there is a path $P_s = (V_s, E_s)$ with $|V_s| \geq 0$ and $|N_{G'}(V_s)| = 0$ in G'. Further assume that the path P_s has $|V_s| < n$. Then there has to be a $v \in V'$ and $v \notin V_s$. Since the graph G' is connected, there has to be a $w \in E_s$ such that a v-w path exists. Hence $N_{G'}(V_s) \geq 1$ and thus contradicting our assumption. The solution P_s forms a Hamiltonian path since $|V_s| = n$.
Since G' is isomorphic to G the NP-hardness results from Hamiltonian Path concerning the graph classes carries over to Long Secluded Path. □

Remark. Note that k could have been any constant value $k \leq n$ in the given reduction. One can also reduce from Long Path without changing G. However the instance constructed $(G', k' = k, \ell = n)$ would have made less of an impact on the parametrized results. This can be useful though if results from Long Path can be helpful.

Proposition 3.5. Long Unsecluded Path *is* NP*-hard on split, bipartite, line and planar graphs even if the parameter* $\ell = 0$.

Proof. We reduce from Long Path. Let (G, k) be an instance of Long Path.
Construction. We construct an instance $(G', k' = k, \ell = 0)$ of Long Unsecluded Path such that G is isomorphic to $G' = (V', E')$. We claim that G is a yes-instance of Long Path if and only if (G', k, ℓ) is a yes-instance of Long Unsecluded Path.
"$\Rightarrow$". Assume there is a longest path $P_p = (V_p, E_p)$ in G. We know that $|V_p| \geq k$ and $|N_G(V_p)| \geq 0$ hence the corresponding path in G' is a long secluded path.
"$\Leftarrow$". Assume there is a long secluded path $P_s = (V_p, E_p)$ in G'. We know that $|V_p| \geq k$ hence the corresponding path in G is a longest path. The graph G' is isomorphic to G, so all the results concerning the graph classes from Long Path are being inherited by Long Unsecluded Path. □

Remark. Since Long Path can be reduced from Hamiltonian Path with the instance $(G', k = n)$ and no change of G', Long Path remains NP-hard on the graph classes where Hamiltonian Path is NP-hard. We could have also reduce from Hamiltonian Path directly without change of G'. However the instance constructed $(G', k' = n, \ell = 0)$ did not seem as promising from a parametrized perspective.

Similarly to our four main path problems, we define four s-t path variants. Two terminals s and t are introduced, which have to be start and endpoint of the path.

Input:	An undirected graph $G = (V, E)$, $s, t \in V$ integers k, ℓ.	
Question:	Is there an s-t path $P = (V_p, E_p)$ in G such that	
	Short Secluded s-t Path:	$\lvert V_p\rvert \leq k, \lvert N_G(V_p)\rvert \leq \ell$?
	Short Unsecluded s-t Path:	$\lvert V_p\rvert \leq k, \lvert N_G(V_p)\rvert \geq \ell$?
	Long Secluded s-t Path:	$\lvert V_p\rvert \geq k, \lvert N_G(V_p)\rvert \leq \ell$?
	Long Unsecluded s-t Path:	$\lvert V_p\rvert \geq k, \lvert N_G(V_p)\rvert \geq \ell$?

One could imagine the problem getting easier in terms of their complexity when a fixed start and endpoint is given, which we were not able to confirm. In fact in some areas, for example when considering solutions for Short Secluded Path, some solutions which were often sufficient are no longer possible. For instance a common solution for Short Secluded Path would be isolated vertices, since they do not have any neighbors and the path is the smallest.

Theorem 3.6. *All four* Short Secluded s-t Path, Short Unsecluded s-t Path, Long Secluded s-t Path, Long Unsecluded s-t Path *are* NP*-complete.*

Interestingly, all four variants reduce to their s-t variants via the same construction. None of the problems were getting easier in terms of their complexity, since it seemed again that the secluded and unsecluded framework was enough to make those problems NP-complete. We are going to present the reduction for Short Secluded s-t Path as an example. However the reduction of the other three path variants work in the same manner.

Proof. We reduce from Short Unsecluded Path, which is shown to be NP-complete in Proposition 3.2. Let (G, k, ℓ) be an instance of Short Unsecluded Path with $|V(G)| = n$.
Construction. We construct an instance $(G', k' = k + 2, \ell' = \ell + 2(\binom{n}{2} - 1))$ of Short Secluded s-t Path. G' is constructed by creating $G_1, G_2, \ldots, G_{\binom{n}{2}}$ which are individually isomorphic to G and joining them together into one graph G'. Two vertices s and t are added to G'. For each combination of two vertices of G one of these subgraphs are taken to connect one of the corresponding pair of vertices to s and the other to t as indicated in Figure 4. We claim that (G, k, ℓ) is a yes-instance of Short Unsecluded Path if and only if (G', k', ℓ') is a yes-instance of Short Secluded s-t Path.
"$\Rightarrow$". Assume there is a path $P_c = (V_c, E_c)$ with $|V_c| = k$ in the graph G. Then there is an s-t path $P_s = (V_s, E_s)$ in G' with $V_s = \{s, v_{c_1}, v_{c_2}, \ldots, v_{c_k}, t\}$ using the corresponding path of P_c in one of the duplicated subgraphs G_i. The resulting path has $|V_s| = k + 2$ and $|N_{G'}(V_s)| = |N_G(V_c)| + 2(\binom{n}{2} - 1)$.
"$\Leftarrow$". Assume there is an s-t path $P_s = (V_s, E_s)$ in G'. The path has to traverse one of the duplicated subgraphs G_i meaning $V_s = \{s, v_{i,c_1}, v_{i,c_2}, \ldots, v_{i,c_k}, t\}$. The corresponding path $P_c = (V_c, E_c)$ in G with $V_c = \{v_{c_1}, v_{c_2}, \ldots, v_{c_k}\}$ can be found by using the corresponding vertices in G from V_c ignoring s and t. Since G_i and G are isomorphic $|N_G(V_c)| = |N_{G'}(V_s)| - (|N_{G'}(s)| + |N_{G'}(t)| - 2) = |N_{G'}(V_s)| - 2(\binom{n}{2} - 1)$ and $|V_c| = |V_s| - 2$. □

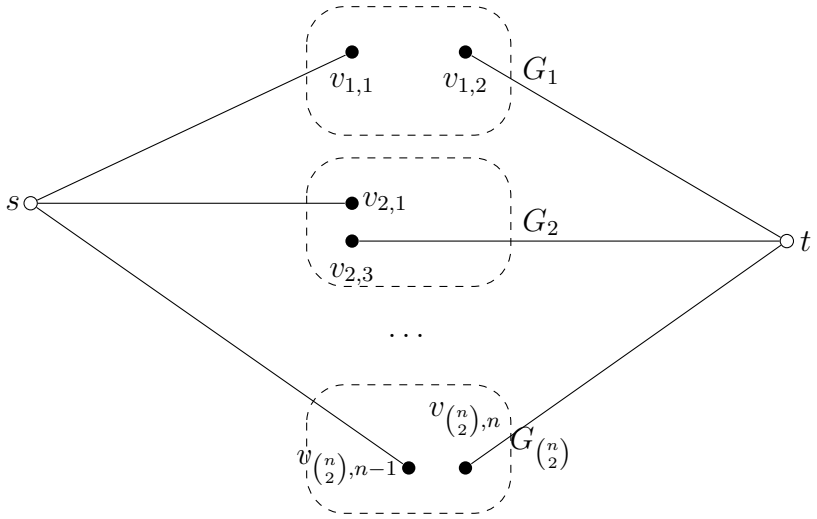

Figure 4: Construction of G' used in the proof of Theorem 3.6

Remark. Notice that the given reduction is also a parametrized reduction with respect to the parameter k, meaning that the hardness results for the parameter k are being inherited from the four main problems.

4 Parametrized Complexity

When looking at the problems from the parametrized point of view, one can first start to see differences between the individual problems. The first assumption that the short variants might actually be easier in terms of their complexity, which was rejected at first, got some support in this area of study. We are going to present our FPT results for the short variants first.

Theorem 4.1. Short Secluded Path *and* Short Unsecluded Path *can be solved in time* $\mathcal{O}(n \cdot \Delta^k)$ *and* Short Unsecluded Path *can be solved in time* $\mathcal{O}(n \cdot \ell^k)$. Short Secluded s-t Path *and* Short Secluded s-t Path *can be solved in time* $\mathcal{O}(\Delta^k)$.

Remark. We are going to show both running times for Short Unsecluded Path, the argumentation for Short Secluded Path will be the same with the difference that the additional bound for Δ does not hold.

Proof. The algorithm, which can solve this problem is a simple breadth-first search. Its worst case running time is $\mathcal{O}(\Delta^k)$ for one vertex. We have to do this for each starting vertex, thus n times. Note that in case of the s-t variants it is only necessary to have the terminal vertex s as a starting vertex. In case of the Short Unsecluded Path a trivial solution can be found if $\Delta \geq \ell$, because then there must exist a single vertex v in the graph G with $|N_G(v)| \geq \ell$, thus being a solution for Short Unsecluded Path. For other cases we then know that $\Delta \leq \ell$ is an upper bound, resulting in the running time $\mathcal{O}(n \cdot \ell^k)$. □

Algorithm 1 used in Lemma 4.2

Input: A graph G, integers k, ℓ
Output: whether the graph has a Short Secluded Path

```
for all S ∈ P(V(G)) with |S| ≤ k do
    if S forms a path & |N_G(S)| ≤ ℓ then
        return true
    end if
end for
return false
```

Even though the other following results do not seem to be promising from a practical point of view, it still shows the differences between the short and the long variants from a parametrized view.

Lemma 4.2. Short Secluded Path *and* Short Unsecluded Path *are contained in* XP *with respect to* k.

Proof. We give Algorithm 1 for any graph $G = (V, E)$ with running time $\mathcal{O}(n^k \cdot m)$ with $n = |V(G)|$ and $m = |E(G)|$. The given algorithm works exemplary for the Short Secluded Path but works for Short Unsecluded Path in the same manner. All possible subsets are checked for being paths and their neighborhoods concerning ℓ. The loop is being traversed $\sum_{i=0}^{k} \binom{n}{i}$ times, due to having to check all possible subsets from $i = 0$ to $i = k$. The internal running time of the loop is $\mathcal{O}(m)$ in order to check if the subset forms a path. An upper bound of the number of subsets is $\binom{n}{k} \cdot 2^k$ and with $\binom{n}{k} \leq \frac{n^k}{k!}$ and $k! > 2^k$ for all $k \geq 3$ resulting in a running time of $\mathcal{O}(n^k \cdot m)$. □

In order to prove W[1] and W[2]-hardness of our problems we first have to get to know a few problems which are W[1] and W[2]-complete.

Clique	
Input:	An undirected graph $G = (V, E)$, integer k.
Question:	Is there a set $C \subseteq V$ with $\|C\| \geq k$ such that there is an edge between every two vertices $c, d \in C$?

Dominating Set	
Input:	An undirected graph $G = (V, E)$, integer k.
Question:	Is there a set $D \subseteq V$ with $\|D\| = k$ such that $N_G[D] = V$?

We will be using Clique and Dominating Set in our parametrized reductions in the following theorems. Clique parametrized by its solution size is known to be W[1]-complete, whereas Dominating Set parametrized by its solution size is W[2]-complete [8].

Theorem 4.3. Short Secluded Path *is* W[1]*-hard with respect to* k.

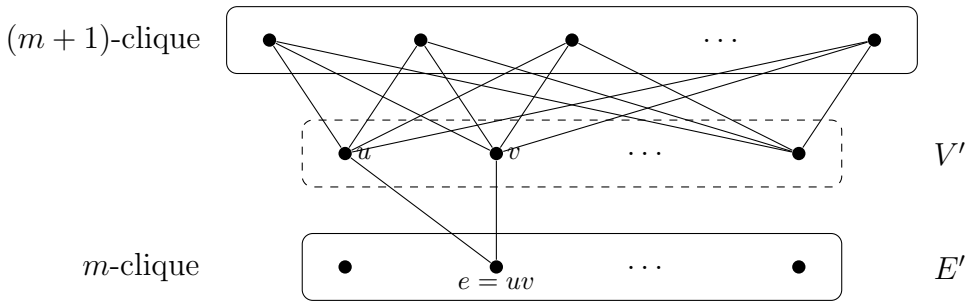

Figure 5: Construction of G' used in the proof of Theorem 4.3

Proof. We reduce from Clique. Let (G, k) be an instance of Clique with $|V(G)| = n$ and $|E(G)| = m$.
Construction. We construct an instance $(G', k' = \binom{k}{2}, \ell = m + k - \binom{k}{2})$ of Short Secluded Path. We refer to Figure 5 for an illustration of G'. The graph G' is constructed by creating an $(m+1)$-clique, where every vertex of the clique is connected to every other vertex by an edge, and n vertices representing $V(G)$, denoted V'. Each vertex from the $(m+1)$-clique got an edge to every vertex in V'. For each edge $e = uv$ we add another vertex e to G' with two additional edges between e and the vertices corresponding to u and v. All the vertices $e_1, e_2, \ldots, e_m \in E'$ created this way are joined together into an m-clique. We claim that (G, k) is a yes-instance of Clique if and only if (G', k', ℓ) is a yes-instance of Short Secluded Path.

"$\Rightarrow$". Assume there is a k-clique $C \subseteq V(G)$ in the graph G. Then there are $\binom{k}{2}$ vertices $v_1, v_2, \ldots, v_{\binom{k}{2}}$ in the part of G' which is corresponding to the edges of the k-clique C. Since E' in G' is a clique there has to be a path P such that $P(V) = \{v_1, v_2, \ldots, v_{\binom{k}{2}}\}$. Observe that $N_{G'}(P) = m - \binom{k}{2} + x$, where x are the number of neighbors of P in V'. Since C is a k-clique $x = k$.
"$\Leftarrow$". Assume there is a path $P_s = (V_s, E_s)$ with $|V_s| \leq \binom{k}{2}$ and $|N_{G'}(V_s)| \leq m + k - \binom{k}{2}$. The path P_s cannot be only one vertex of V', because its neighborhood would be $N_{G'}(P_s) \geq m + 1$, which is too much for values $k \geq 3$. The path P_s cannot include any vertices from either the $(m+1)$-clique or V', because if it would contain a vertex from the $(m+1)$-clique then $N_{G'}(P_s) \geq n + m + 1 - |V(P_s)|$. But $N_{G'}(P_s) \leq m + k - \binom{k}{2}$, which can only be true if:

$$n + m + 1 - |V(P_s)| = m + k - \binom{k}{2}$$

$$\Leftrightarrow \qquad 1 = \underbrace{|V(P_s)|}_{\leq \binom{k}{2}} + \underbrace{(k - n)}_{\leq 0} - \binom{k}{2}.$$

Hence $P_s \subseteq E'$. It follows $|N_{G'}(P_s)| = m - |V(P_s)| + |N_{G'}(P_s) \cap V'|$ and thus,

because $|V(P_s)| \leq \binom{k}{2}$:

$$m - \binom{k}{2} + |N_{G'}(P_s) \cap V'| \leq |N_{G'}(P_s)| \leq m + k - \binom{k}{2}$$

Meaning $|N_{G'}(P_s) \cap V'| \leq k$. The set of vertices in the path $V(P_s)$ is the smallest, if the corresponding vertices of $V' \cap N_{G'}(P_s)$ are a clique in G, because a clique has the best ratio between edges and vertices, meaning $|V(P_s)| \leq \binom{|N_{G'}(P_s) \cap V'|}{2}$

$$m - \binom{|N_{G'}(P_s) \cap V'|}{2} + |N_{G'}(P_s) \cap V'| \leq |N_{G'}(P_s)| \leq m + k - \binom{k}{2}$$

$$\Leftrightarrow \qquad |N_{G'}(P_s) \cap V'| - k \leq \binom{|N_{G'}(P_s) \cap V'|}{2} - \binom{k}{2}$$

Which is true for values $k \geq 3$ and only $k \geq |N_{G'}(P_s) \cap V'|$, meaning that the path has to use its allowed length of $\binom{k}{2} - 1$. The clique $C \subseteq G$ can by found by selecting the corresponding vertices of $V' \cap N_{G'}(P_s)$. □

Theorem 4.4. Short Unsecluded Path *and* Long Unsecluded Path *is* W[2]*-hard with respect to* k*.*

We show the reduction for Short Unsecluded Path in detail, however note that the reduction for Long Unsecluded Path works in the same manner.

Proof. We reduce from Dominating Set. Let (G, k) be an instance of Dominating Set with $|V(G)| = n$.
Construction. We construct an instance $(G', k' = 2k+1, \ell = n^2+2n+n^2k-k)$ of Short Unsecluded Path. We construct G' by creating two replicas of $V(G)$ denoted V' and V'' and $k+1$ vertices each with n isolated vertices as neighbors called K. For each vertex $v \in V(G)$ we add an edge $v'v''$ between $v' \in V'$ and $v'' \in V''$ in G'. For each edge $e = vw \in E(G)$ we create two additional edges in G' namely $v'w''$ and $w'v''$. The resulting graph G' is illustrated in Figure 6. We claim that (G, k) is a yes-instance of Dominating Set if and only if (G', k', ℓ) is a yes-instance of Short Unsecluded Path.

"⇒". Assume there is a dominating set $D = \{d_1, d_2, \ldots, d_k\} \subseteq V(G)$ in the graph G. We construct a path P_s in G' with $p_i \in P$ such that $V(P_s) = \{p_1, d'_1, p_2, \ldots, d'_{k-1}, p_k, d'_k, p_{k+1}\}$, which can always be done, because every vertex of V' is connected to every vertex of P. The resulting path is a short unsecluded path with $|V(P_s)| = 2k + 1$. Furthermore $|N_{G'}(P_s) \cap V''| = n$ since D is a dominating set. Since every vertex in K is in the neighborhood due to $P \subset V(P_s)$, the resulting neighborhood of the path is $|N_{G'}(P_s)| = n^2 \cdot (k + 1) + (n - k) + n = n^2 + 2n + n^2k - k$.
"⇐". Assume there is a path P_s with $|V(P_s)| \leq 2k + 1$ and $|N_{G'}(P_s)| \geq n^2 + 2n + n^2k - k$ in G'. It has to hold $P \subset V(P_s)$, because imagine that there

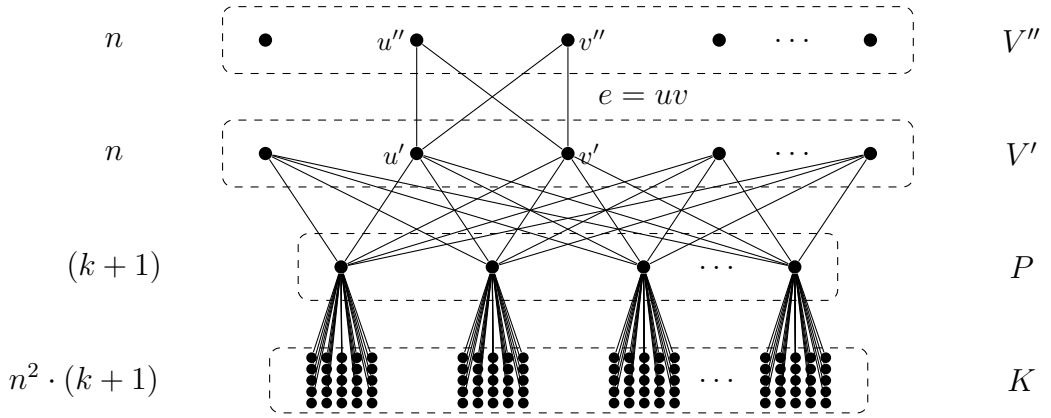

Figure 6: Construction of G' used in the proof of Theorem 4.4

is a $p \in P$ with $p \notin P_s$. Then the neighborhood is bounded by all the remaining vertices: $|N_{G'}(P_s)| \leq 2n + k + 1 + n^2k$, resulting in:

$$2n + k + 1 + n^2k \geq |N_{G'}(P_s)| \geq n^2 + 2n + n^2k - k$$

$$\Rightarrow \qquad 2k + 1 \geq n^2.$$

Since k is bounded by n, this can never be true for $n > 2$. Thus $P \subset V(P_s)$. In order to achieve that, it has to hold $|V(P_s) \cap V'| \geq k$, because the vertices in P do not have any other edges apart from the ones connected to every other vertex in V'. Furthermore it has to hold $|V(P_s) \cap V'| = k$, because there are only $n^2 + n^2k + 2n - k$ vertices left from which all of them have to be in the neighborhood and cannot be included in the path. To gain the remaining n vertices from V'' in $N_{G'}(P_s)$ the corresponding vertices from $V(P_s) \cap V'$ have to form a dominating set in G. □

Remark. For Long Unsecluded Path the argumentation and reduction would be the same. Only the part where it is discussed that the path could be shorter than $2k + 1$ would have been left out.

5 Graph Classes

Since our problems are, as seen in the given reductions in Section 3, closely related to Hamiltonian Path, we hypothesized that our new path problems are NP-complete and polynomial-time solvable on the same graph classes as Hamiltonian Path. This assessment was validated at least on the graph classes that we checked. As seen in Section 3, they inherit the hardness results of Hamiltonian Path, meaning split, bipartite, line and planar graphs. Note that on some graph classes like complete graphs our problems are rather trivial, due to the property of being able to easily construct paths and check their neighborhood. More interesting graph classes are covered in this section and the corresponding algorithms are shown.

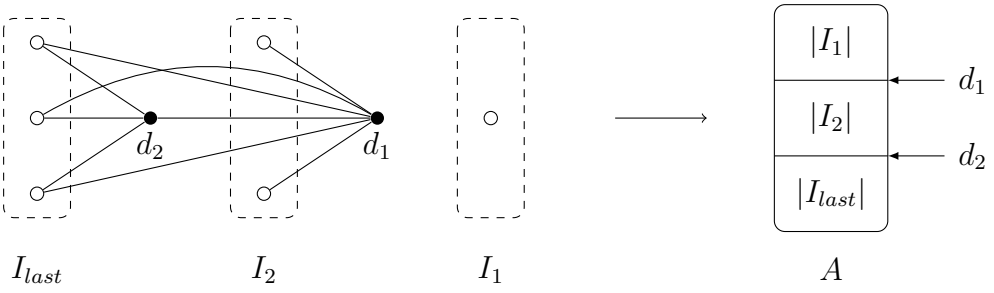

Figure 7: A threshold graph with its corresponding array representation. The vertices added as dominating vertices are filled.

Trees. It is relatively simple to solve any of the eight problem variants on a tree T due to its properties. There is only one unique x-y path connecting two terminal vertices $x, y \in V(T)$ [6]. A simple brute force algorithm will suffice for a polynomial-time algorithm. Simply take all possible n^2 combinations of start and endpoints. Since the path connecting those is unique, one only has to check the neighborhood of that path against the parameter ℓ. For the four s-t variants it is even simpler. It is only necessary to check the neighborhood of the one s-t path, which connects both terminal vertices.

5.1 Threshold Graphs

The class of threshold graphs can be defined recursively with the following rules:

1. The graph $G = (V, E)$ with $|V| = 1$ and $E = \emptyset$ is a threshold graph.
2. Adding an isolated vertex to a threshold graph, is also a threshold graph.
3. Adding a dominating vertex to a threshold graph, is also a threshold graph.

Only graphs which can be constructed with these rules are threshold graphs [19]. We use an array representation of the graph for most of our problems, which stands for a way the graph could have been created. The array stores the information how many isolated vertices were added before the next dominating vertex. The array A can be constructed by Algorithm 2. Observe that the deconstruction begins at $A[1]$ to $A[a_{last}]$, so in order to reconstruct the graph one has to traverse the array in reverse order, that is from $A[a_{last}]$ to $A[1]$. An example of a threshold graph and its array are given in Figure 7.

Theorem 5.1. Short Unsecluded Path *on threshold graphs can be solved in time* $\mathcal{O}(n + m)$.

Proof. Given an instance of Short Unsecluded Path (G, k, ℓ), there is only one path to consider as the solution on a given threshold graph $G = (V, E)$. If

Algorithm 2 Deconstruction of a threshold graph

Input: A threshold graph G
Output: array representation A of the threshold graph G

```
1: i ← 1
2: G' ← G
3: while G' still contains at least one vertex do
4:    Let I be the set of isolated vertices in G'
5:    A[i] ← |I|
6:    G' ← G' \ I
7:    Let v be a vertex of maximal degree
8:    G' ← G' \ {v}
9:    i ← i + 1
10: end while
11: return graphArray
```

the graph G is not connected, that means that there are i isolated vertices with $I = \{i \mid i \text{ is isolated}\}$ which are not candidates for a short secluded path and thus can be ignored. The remaining graph G' is connected, since there cannot be more than one connected component with more than one vertex. It then follows from the definition of threshold graphs that there is a vertex v, which has $N_G(v) = V \setminus (I \cup \{v\})$. There is no better candidate solution than the path $P = (V_p, E_p)$ with $V_p = \{v\}$, $E = \emptyset$ and $|N_G(v)| = n - 1 - |I|$, because the neighborhood can not get any bigger than $n-1-|I|$ and the length of the path can not get smaller than 0. All the algorithm has to do is count the number of isolated vertices which takes $\mathcal{O}(n+m)$ time and then decide if $\ell \geq n - 1 - |I|$ holds, which takes constant computational effort. □

Lemma 5.2. *The array representation of a threshold graph can be constructed in time* $\mathcal{O}(n^2)$.

Proof. The algorithm for deconstruction of a threshold graph can be found Algorithm 2. The loop is at most traversed n times as in each iteration at least one vertex is deleted from the graph. The running time inside a loop is $\mathcal{O}(n)$. □

Remark. For the following proofs we will also need a function $d(i) = \sum_{j=i+1}^{A_{last}} A[j]$ which stand for the number of vertices (originally added as isolated ones), which are dominated by the ith dominating vertex.

Theorem 5.3. Long Unsecluded Path *on threshold graphs can be solved in time* $\mathcal{O}(n^2)$.

Proof. We have already shown in Theorem 5.1, that there is one vertex d_1 which has the all remaining vertices, apart from the set of isolated vertices I_1, in its neighborhood $|N_G(d_1)| = n - 1 - |I|$. Starting from that vertex the Algorithm 3 is going to let the path grow in length by one vertex while their

neighborhood is reduced by one vertex, since all remaining vertices are already in the neighborhood. Between each two vertex d_s and d_r added as a dominating one, we try to add a vertex $j_{s+1} \in J_{s+1}$ where $J_i = \bigcup_{j=i}^{last} I_j$. The resulting path $P = (V, E)$ will be $V = \{j_2, d_1, j_3, d_2, \ldots, d_{last-1}, j_{last}\}$ alternating between vertices added as dominating and isolated vertices. Note that if $J_i = \emptyset$ then j_i is left out instead. The loop in Line 3 of Algorithm 3 is traversed at most n times, because the length of the array cannot exceed n. The calculation of d is also bound by n and together with the other operations inside the loop which take constant effort the resulting running time is $\mathcal{O}(n^2)$. □

Algorithm 3 LONG UNSECLUDED PATH on threshold graphs

Input: array representation of the threshold graph A, the two integers k, ℓ and the function $d(i) = \sum_{j=i+1}^{A_{last}} A[j]$

Output: whether there is a LONG UNSECLUDED PATH in the graph

```
1:  pathLength ← 1
2:  neighborhood ← n − A[1] − 1
3:  for i ← 1 to length(A) − 1 do
4:    // adding an isolated vertex to the path if possible:
5:    if d(i) ≥ 0 then
6:      pathLength ← pathLength +1
7:      neighborhood ← neighborhood −1
8:      if pathLength ≥ k and neighborhood ≥ ℓ then
9:        return true
10:     end if
11:   end if
12:   // adding a dominating vertex to the path:
13:   pathLength ← pathLength +1
14:   neighborhood ← neighborhood −1
15:   if pathLength ≥ k and neighborhood ≥ ℓ then
16:     return true
17:   end if
18: end for
19: return false
```

5.2 Cographs

Since we were able to show that the problems are polynomial-time solvable on threshold graphs, we are going to show that the same is true on a more general graph class. An introduction to cographs is given by Corneil, Lerchs, and Burlingham [5]. Each cograph can be represented by a cotree which, similarly to the array representation of the threshold graphs, does represent the way the cograph can be constructed. In this section we will refer to the vertices of the cotree as *nodes* to avoid confusion with the vertices of the cograph. A cotree for a graph G is a tree with its leafs representing the vertices of G. The internal nodes

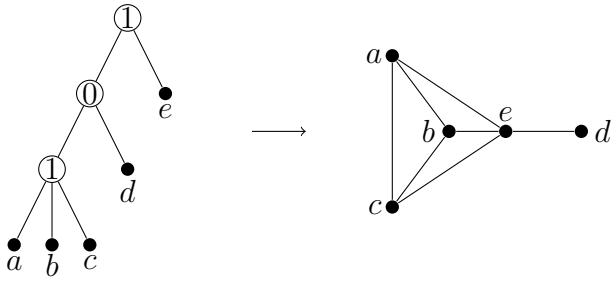

Figure 8: Example of a cograph and its cotree representation

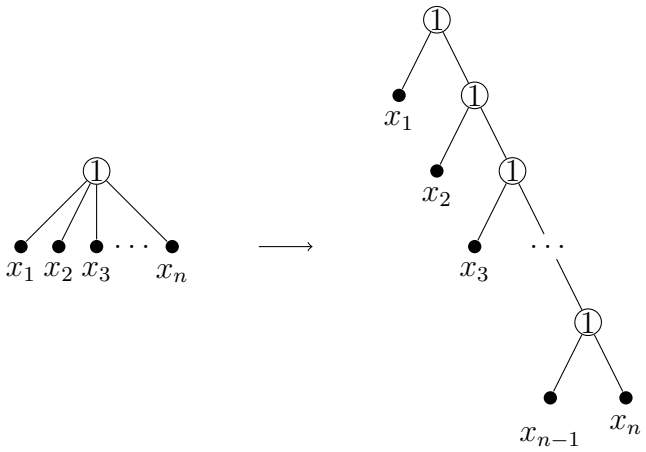

Figure 9: Example of binarizing a node of a cotree

of the tree are labeled 0 or 1. Two vertices $x, y \in V(G)$ are adjacent if and only if the least common ancestor of x and y is labeled 1 [26]. The root of the cotree is always a 1-labeled node and the graph is disconnected if and only if it has only one child [16]. As seen in Figure 8, it is quite simple to construct a cograph given a cotree representation. When working with cographs it can be useful to use a binarized version of the cotree illustrated in Figure 9, which was also used by Lin, Olariu, and Schwing [17]. It was recently discovered that finding the longest path on a cograph G can be done in polynomial time [20], which we are going to use in some of our algorithms, denoted as $getLongestPath(G)$.

Theorem 5.4. Long Secluded Path *on cographs can be solved in polynomial time.*

Proof. We are first going to create an array with all of the best possible neighborhoods concerning the secluded framework, meaning of two paths with identical length we always prefer the one with a smaller neighborhood. For an overview of the algorithm we refer to Algorithm 4. It is recursively defined, thus in order to get the proper array for the whole graph the root of the cotree has to be the input. For Long Secluded Path the array A is then to be

Algorithm 4 Recursively defined function **leastNb** for Theorem 5.4

Input: graph G, node t of the cotree

Output: array A_N representing the least neighbors for all sizes of paths on the subtree T with t as its root

1: **if** $type(t) = leaf$ **then**
2: $A_N[1] \leftarrow 0$
3: **else**
4: $left \leftarrow leftChild(t)$
5: $right \leftarrow rightChild(t)$
6: **end if**
7: **if** $type(t) = 0$ **then**
8: **for** $i \leftarrow 1$ **to** $length(N)$ **do**
9: $A_N[i] \leftarrow \min(\textbf{leastNb}(left)[i], \textbf{leastNb}(right)[i])$
10: **end for**
11: **end if**
12: **if** $type(t) = 1$ **then**
13: **for** $i \leftarrow 1$ **to** $length(N)$ **do**
14: $bestLeft \leftarrow \textbf{leastNb}(left)[i] + \textbf{numberOfVertices}(right)$
15: $bestRight \leftarrow \textbf{leastNb}(right)[i] + \textbf{numberOfVertices}(left)$
16: $A_N[i] \leftarrow \min(bestLeft, bestRight)$
17: $L \leftarrow \textbf{getLongestPath}(\mathrm{T})$
18: $A_N[|V(L)|] \leftarrow |N_G(V(L))|$
19: **end for**
20: **end if**
21: **return** A_N

checked whether there is an $A[x] = y$ with $x \geq k$ and $y \leq \ell$.
Since there are three types of nodes the algorithm distinguishes between those. If the node t of the cotree is a leaf, then the best path consists obviously of 1 vertex and its neighborhood is 0. If the node t is labeled with a 0 as in Line 7 of Algorithm 4, then there are no vertices between the subtrees T_{left} and T_{right}. Meaning that the paths do not change. Only the two arrays have to be merged together. If the node t is labeled with a 1 as in Line 13 of Algorithm 4, then there is an edge between any vertex of T_{left} and any vertex of T_{right}. Thus there are three possibilities where the best paths can occur. Either in T_{left} (Line 14), in T_{right} (Line 15) or in both. If the path P has a vertex in both T_{left} and T_{right} then its neighborhood is $N_G(V(P)) = (T_{left} \cup T_{right}) \setminus V(P)$. Hence the best path, in terms of its neighborhood, one can acquire is the longest path in $T_{left} \cup T_{right}$. We use the algorithm given by [20] to find that path in Line 17 of Algorithm 4.
In the worst case the cotree consists of only nodes labeled with a 1 and leafs. Since the subtree T_{sub} of the next function call does not contain t itself, it the function gets called at most n times. With the running time of finding the longest path is polynomial the overall running time also stays polynomial. □

6 Conclusion and further work

Generally speaking we can conclude that all of our eight variations seem to be closely related to Hamiltonian Path in terms of its complexity. Apart from filling in the gaps in terms of parametrized complexity and the complexity on the graph classes a few other things appear to be of further interest. The problem that stands out from this observation is Short Unsecluded Path. Not only were we able to give an FPT algorithm in respect to k and l but the reduction used in Section 3 suggests that, since a lot of adjacent vertices were added to each vertex, the problem might be also efficiently solvable on graphs with bounded degree. Those graphs can be especially interesting in area of real world applications, where the degree of single vertex should not exceed a certain threshold.

Literature

[1] C. Benzaken, P. Hammer, and D. de Werra. "Split graphs of Dilworth number 2". In: *Discrete Mathematics* 55.2 (1985), pp. 123–127 (cit. on p. 9).

[2] R. van Bevern, T. Fluschnik, G. B. Mertzios, H. Molter, M. Sorge, and O. Suchý. "Finding Secluded Places of Special Interest in Graphs". In: *Proceedings of the 11th International Symposium on Parameterized and Exact Computation (IPEC'16)*. Vol. 63. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, 5:1–5:16 (cit. on p. 7).

[3] H. L. Bodlaender. "A tourist guide through treewidth". In: *Developments in Theoretical Computer Science* 1 (1994) (cit. on p. 9).

[4] S. Chechik, M. P. Johnson, M. Parter, and D. Peleg. "Secluded Connectivity Problems". In: *Proceedings of the 21st Annual European Symposium on Algorithms (ESA'13)*. Vol. 8125. Lecture Notes in Computer Science. Springer, 2013, pp. 301–312 (cit. on p. 7).

[5] D. Corneil, H. Lerchs, and L. Burlingham. "Complement reducible graphs". In: *Discrete Applied Mathematics* (1981), pp. 163–174 (cit. on p. 22).

[6] R. Diestel. *Graph Theory*. Springer, 2016 (cit. on pp. 8, 20).

[7] E. W. Dijkstra. "A Note on Two Problems in Connexion with Graphs". In: *Numerische Mathematik* 1.1 (1959), pp. 269–271 (cit. on p. 6).

[8] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1998 (cit. on pp. 10, 16).

[9] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006 (cit. on p. 10).

[10] F. V. Fomin, P. A. Golovach, and J. H. Korhonen. "On the Parameterized Complexity of Cutting a Few Vertices from a Graph". In: *Proceedings of the 38th International Symposium on Mathematical Foundations of Computer Science (MFCS'13)*. Vol. 8087. Lecture Notes in Computer Science. Springer, 2013, pp. 421–432 (cit. on p. 7).

[11] F. V. Fomin, P. A. Golovach, N. Karpov, and A. S. Kulikov. "Parameterized Complexity of Secluded Connectivity Problems". In: *Proceedings of the 35th Annual Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS'15)*. Vol. 45. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pp. 408–419 (cit. on p. 7).

[12] F. Harary. *Graph Theory*. Addison-Wesley, 1969 (cit. on p. 10).

[13] R. M. Karp. "Reducibility Among Combinatorial Problems". In: *Proceedings of a Symposium on the Complexity of Computer Computations*. The IBM Research Symposia Series. Plenum Press, New York, 1972, pp. 85–103 (cit. on p. 11).

[14] M. S. Krishnamoorthy. “An NP-hard Problem in Bipartite Graphs”. In: *SIGACT News* 7.1 (1975), pp. 26–26 (cit. on p. 11).

[15] T. Lai and S. Wei. “The Edge Hamiltonian Path Problem is NP-Complete for Bipartite Graphs”. In: *Information Processing Letters* 46.1 (1993), pp. 21–26 (cit. on p. 11).

[16] R. Lin, S. Olariu, and G. Pruesse. “An optimal path cover algorithm for cographs”. In: *Computers and Mathematics with Applications* (1995), pp. 75–83 (cit. on p. 23).

[17] R. Lin, S. Olariu, and J. Schwing. “An efficient EREW algorithm for minimum path cover and Hamiltonicity on cographs”. In: *Parallel Algorithms and Applications* 2.1-2 (1994), pp. 99–113 (cit. on p. 23).

[18] D. S. J. M. R. Garey and R. E. Tarjan. “The Planar Hamiltonian Circuit Problem is NP-Complete”. In: *SIAM Journal on Computing* 5.4 (1975), pp. 704–714 (cit. on p. 11).

[19] N. Mahadev and U. Peled. *Threshold Graphs and Related Topics*. Elsevier, 1995 (cit. on p. 20).

[20] G. B. Mertzios and D. G. Corneil. “A Simple Polynomial Algorithm for the Longest Path Problem on Cocomparability Graphs”. In: *SIAM Journal on Discrete Mathematics* 26.3 (2012), pp. 940–963 (cit. on pp. 23, 25).

[21] S. Mondal, M. Pal, and T. K. Pal. “An Optimal Algorithm to Solve the All-Pairs Shortest Paths Problem on Permutation Graphs”. In: *Journal of Mathematical Modelling and Algorithms* 2.1 (2003), pp. 57–65 (cit. on p. 10).

[22] B. Monien. “How to Find Long Paths Efficiently”. In: *North-Holland Mathematics Studies* 109 (1985), pp. 239–254 (cit. on p. 11).

[23] H. Müller. “Hamiltonian circuits in chordal bipartite graphs”. In: *Discrete Mathematics* 156.1 (1996), pp. 291–298 (cit. on p. 11).

[24] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006 (cit. on p. 10).

[25] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2003 (cit. on pp. 9, 10).

[26] J. P. Spinrad. *Efficient Graph Representations*. Fields Institute Monographs. American Mathematical Society, 2003 (cit. on p. 23).