



**HAL**  
open science

# The Dynamic Steiner Tree problem: definitions, complexity, algorithms

Stefan Balev, Yoann Pigné, Eric Sanlaville, Mathilde Vernet

## ► To cite this version:

Stefan Balev, Yoann Pigné, Eric Sanlaville, Mathilde Vernet. The Dynamic Steiner Tree problem: definitions, complexity, algorithms. 2024. hal-04735967

**HAL Id: hal-04735967**

**<https://normandie-univ.hal.science/hal-04735967v1>**

Preprint submitted on 14 Oct 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# 1 The Dynamic Steiner Tree problem: definitions, 2 complexity, algorithms

3 Stefan Balev

4 Yoann Pigné

5 **Éric Sanlaville**<sup>1</sup>

6 Université Le Havre Normandie, Univ Rouen Normandie, INSA Rouen Normandie, Normandie

7 Univ, LITIS UR 4108, F-76600 Le Havre, France

8 **Mathilde Vernet**

9 LIA, Avignon Université, Avignon, France

## 10 — Abstract —

---

11 This paper introduces an extension of the Steiner tree problem applied to dynamic graphs. In  
12 various application domains of this problem, the associated graphs undergo temporal changes, such  
13 as variations in the edge set or associated costs. The paper presents three extension models, and  
14 we opt for the one demonstrating the most desirable features: a fixed Steiner set (a set of selected  
15 vertices) maintained throughout the time horizon and of minimum cardinality, ensuring connectivity  
16 among the terminal vertices.

17 We show that the resulting problem of minimizing the size of the Steiner set is NP-hard, even  
18 when dealing with only two terminals [and a lifetime of 2 time steps](#). This contrasts with the  
19 corresponding static problem, which is among the rare polynomial versions of the Steiner tree  
20 problem. However, polynomial algorithms exist when the size of the Steiner set is bounded. An  
21 algorithm is designed, analyzed and tested on specially generated and on real data sets. We show  
22 that it solves exactly non trivial instances when the Steiner set size is bounded.

23 **2012 ACM Subject Classification** Mathematics of computing → Paths and connectivity problems

24 **Keywords and phrases** Steiner Tree, Dynamic Graph, Complexity, experimental study

25 **Funding** This work was Supported by the French ANR, project ANR-22-CE48-0001 (TEMPOGRAL)

26

## 27 **1 Introduction and contributions of this work**

28 The Steiner tree problem is one of the most studied combinatorial problems on graphs.  
29 Roughly speaking, the goal is to preserve the connectivity among some selected vertices,  
30 called terminals, minimizing the cost associated to the chosen edges, or vertices, in the  
31 process. This kind of problems occurs in many application domains, as in communication  
32 networks, sensor networks, social networks, logistic networks, etc.

33 Although it is simple to formulate, there is no easy way to solve it as it was very early  
34 proven to be NP-hard [15]. Furthermore, the problem remains NP-hard even in most of  
35 its simple versions. Indeed, it is the case even with unitary costs on the edges[12], that is,  
36 when the objective is to minimize the number of vertices to connect the terminal vertices. In  
37 the following, the vertex set used to connect the terminals, including the terminal vertices  
38 themselves, will be called *the Steiner set*.

39 However, there exist cases for which the Steiner problem is easy as it can be solved using  
40 polynomial algorithms. The first case is when the number of terminals is two, as it then

---

<sup>1</sup> corresponding author

41 reduces to a shortest path problem. The second case is when every vertex of the graph is a  
 42 terminal as it then reduces to a spanning tree problem.

43 In the application domains cited above, the underlying graph may change over time.  
 44 This is undoubtedly the case for social networks as interactions among people have a limited  
 45 lifespan. This is also true for some communication networks (for instance ad-hoc networks or  
 46 mobile networks). And also for logistic networks, even if the dynamic is much slower, because  
 47 of works on the road or congestion phenomena for instance. For some years now, a growing  
 48 literature has been considering the well-known classical combinatorial problems in this new  
 49 setting. Recalling all papers dealing with dynamic graphs (the name may vary) is out of the  
 50 scope of this paper, but one may cite, as the most relevant for this work, [22, 17] for paths and  
 51 their extensions like hamiltonian cycles, [11, 18, 19] for flows, [6, 20] for some considerations  
 52 about connectivity issues, ... Observe that there are different ways (and names) to define a  
 53 dynamic graph (temporal graphs, dynamic or temporal networks, or time evolving graphs).  
 54 Still, and when time is considered as a discrete variable, such a graph is basically constituted  
 55 of an ordered sequence of graphs indexed by time:  $\mathcal{G} = (G_i)_{i \in \mathcal{T}}$ . This is the way it will be  
 56 considered throughout the paper. The term of dynamic graph is preferred because, as we  
 57 shall see, it is not mandatory to know the graph evolution in advance to find a *Steiner set*  
 58 (the set of vertices used to connect the terminals).

59 Designing the temporal counterpart of a given optimization problem on static graphs  
 60 may not always be easy. This is specially the case when the problem involves connectivity  
 61 issues, as connectivity in a temporal setting may have different meanings. A Steiner tree  
 62 is the cheapest way to ensure connectivity between a subset of nodes called terminals. In  
 63 a temporal setting, we are looking for a structure that maintains connectivity whereas the  
 64 graph is changing. Basically, connectivity here may be defined in two ways. In the first  
 65 way, one simply looks for the existence of a journey, also called temporal path, from each  
 66 terminal to any other terminal. This journey-based connectivity ensures some information,  
 67 or some good, will eventually be transferred. But this is a one shot property: typically, after  
 68 a given time, no journey may exist for a couple of terminals and the transfer is no more  
 69 possible. Conversely, in the second way one may wish to maintain some "instantaneous"  
 70 connectivity: at each time, a path exists between each pair of terminals. *We claim that*  
 71 *instantaneous, or path-based connectivity (see [20]), may be better adapted to some situations,*  
 72 *like a set of mobile robots that cooperate for a given common goal, thus needing frequent*  
 73 *communications between some distinguished nodes of the network. Many works exist that*  
 74 *study the existence of journeys, see for instance the seminal work of [22]. Many works also*  
 75 *study some journey-based extensions of [connected components](#), see [4, 3], or of spanning trees,*  
 76 *namely spanners, see [2, 7]. The path-based extensions of spanning tree are of less interest,*  
 77 *as it may consist either in computing the minimum spanning tree at each snapshot (without*  
 78 *building any persistent structure), or in computing the spanning tree of the intersection of*  
 79 *all snapshots (using only edges constantly present). Note that to the best of our knowledge,*  
 80 *no previous work considers the solving of direct extensions of Steiner trees. [Still, the recent](#)*  
 81 *[work of Klobas et al \[16\] introduced a journey-based temporal version : when does it exist](#)*  
 82 *[a journey between all terminal vertices? However, their paper studies the dual problem](#)*  
 83 *[of finding a minimum temporal labelling allowing the existence of a temporal Steiner tree.](#)*  
 84 *[Although this problem is NP-Hard, The number of labels is bounded by  \$2n - 4\$ , and these](#)*  
 85 *[labels may be assigned to a tree whose leafs are the terminals, plus one edge. When the](#)*  
 86 *[temporal labelling is given, we shall see that the number of edges \(and of vertices\) to ensure](#)*  
 87 *[instantaneous connectivity among the terminals may be much larger.](#)*

88 This paper first investigates the temporal counterpart of Steiner tree problem when the

89 goal is to maintain instantaneous connectivity at minimal cost. Several possible extensions of  
90 the Steiner Tree problem to the case of dynamic graphs are presented. Note that the study is  
91 restricted to the case of unit costs. The obvious extension is to compute the Steiner trees for  
92 all time steps. Its drawbacks are that one must keep a different set of intermediate vertices  
93 at each time step, which is not of practical use; and it is computationally intensive. The  
94 other proposed extensions build a *Steiner set*, that is, a subset of vertices that will guarantee  
95 the connectivity of the terminal vertices throughout the lifetime of the graph. The second  
96 extension builds at each time step a spanning tree whose vertices are the terminals plus all  
97 the vertices of the Steiner set. We show that this problem cannot be reduced to successive  
98 solving of classical Steiner tree problems. However, the drawback of this model is that it  
99 retains too many edges that are not necessary at each time step to keep the terminal vertices  
100 connected. Therefore, the total cost might exceed by far the cost for the other models. *Our*  
101 *last extension minimizes the size of the Steiner set*. Note that the sum of the costs of the  
102 edges used at each time step might be used as a second criterion in case of arbitrary costs.  
103 This definition is more suitable for the applications, as the obtained solutions will be less  
104 resource consuming in terms of vertices and edges used, and is used in the remaining of the  
105 paper. However, we simply prove that *the associated decision problem is NP-Complete even*  
106 *when the costs are not considered, and there are only two terminals*. [A more intricated proof](#)  
107 [shows it remains the case even with two time steps](#).

108 This calls for the design of exact algorithms that should be efficient for Steiner sets of  
109 small size, or of approximation algorithms, that will be designed specifically for the dynamic  
110 setting. *This paper proposes an exact algorithm* that computes all Steiner sets of a given size,  
111 studies its complexity, and provides some experimental tests. The algorithm might be used  
112 on-line: at each time step, all vertex sets that are so far Steiner sets, are computed. [It might](#)  
113 [be worth to discuss here about the distinction between temporal or dynamic graphs and](#)  
114 [multi-layered graphs](#). Multi-layered graphs have been studied in a number of papers, see for  
115 instance [5] where several kinds of subgraph searches (including connected components) are  
116 investigated. They are similar to temporal graphs in the sense that they can also be defined  
117 as a collection of static graphs with an identical set of vertices. However, these static graphs  
118 are not ordered, contrary to the snapshots of temporal graphs. As we shall see in the next  
119 section, definitions of Steiner problems based on maintaining instantaneous connectivity do  
120 not need to consider some ordering of the snapshots. However, it is interesting to solve these  
121 problems through on-line algorithms, so that at each time step, the collection of current  
122 Steiner sets is given. That is why we introduced these extensions of the Steiner problems  
123 through the setting of dynamic graphs (to our knowledge, there is no paper generalizing the  
124 Steiner problem to multi-layered graphs).

125 Experiments with different randomly generated dynamic graphs show that *our algorithm*  
126 *is able to solve medium size instances*. The number of solutions is highly dependent on the  
127 type of graphs. The results confirm that Steiner sets are much more numerous on scale-free  
128 graphs. The algorithm is also applied to a dynamic graph obtained from real data about  
129 mobile ad hoc networks. The graph is small, but the time horizon is large. *The algorithm is*  
130 *able to find rapidly all Steiner sets, for different parameter values*.

131 The main contributions of this paper are therefore:

- 132 ■ We discuss how to extend the Steiner tree problem to dynamic graphs. Among the  
133 possible extensions, we identify the more relevant one.
- 134 ■ We show that unlike its static counterpart, the dynamic Steiner problem is NP-hard even  
135 with two terminals.
- 136 ■ We propose an exact algorithm that computes all Steiner sets of a given size, study its

137 complexity and test it experimentally on generated and real-word instances.

## 138 **2 The Dynamic Steiner Set problem**

139 Let us first recall the static problem settings. Let  $G = (V, E)$  be a (static) graph. A non-  
 140 negative weight  $w_e$  is associated to each edge  $e \in E$ . For a given subset of vertices  $S \subset V$ ,  
 141 called *terminals*, the objective is to find a tree of minimum weight covering all vertices of  $S$ .  
 142 If the case of unit costs, It is of course equivalent to minimize the number of vertices of the  
 143 tree.

144 Let us see now how the Steiner Tree Problem can be extended to dynamic graphs. Let  $\mathcal{G}$   
 145 be an non-directed dynamic graph such that  $\mathcal{G} = (G_1, \dots, G_L)$ , with  $G_i = (V, E_i) \forall 1 \leq i \leq L$   
 146 and  $E = \bigcup_i E_i$ . The successive  $G_i$ s are called *snapshots* of  $\mathcal{G}$  at each time step. As in the  
 147 static case, let us consider a subset  $S$  of special vertices, called terminals. The goal is still to  
 148 ensure connectivity between the terminals. Note that we limit our study to the case where  
 149 no travel time is associated to the edges. Hence there is no travel time associated to paths  
 150 either: the (instantaneous) connectivity requirement applies to each time step.

### 151 **2.1 Direct extension: allowing a varying Steiner set**

152 The direct way to extend the Steiner Tree problem to such dynamic graphs is of course to  
 153 compute, at each time step, the Steiner tree associated to  $S$ . This extension has one major  
 154 drawback: at each time step, the Steiner set is different. It is not really convenient, for  
 155 instance in a communication network, to change the intermediate nodes at each time step.  
 156 More deeply, we do not make any use of the knowledge of the dynamic graph as a whole,  
 157 considering each  $G_i$  separately. Note also that this approach is very time consuming, as a  
 158 complete Steiner tree is recomputed at each time step. Hence in the remaining of the section,  
 159 we focus on computing a Steiner set that is fixed during all the time of study.

### 160 **2.2 Second model: Fully Connected Minimum Steiner Set**

#### 161 **2.2.1 Definition**

162 **► Definition 1** (Fully Connected Minimum Steiner Set). *Let us consider some dynamic graph*  
 163  $\mathcal{G}$ . *For a given vertex set  $S \subset V$  of terminals, find  $E'_i \subset E_i \forall i \leq L$  and  $V'$  with  $S \subset V' \subset V$*   
 164 *such that:*

- 165 **■**  $G'_i = (V', E'_i)$  *is a connected graph  $\forall i \leq L$*
- 166 **■**  $|V'|$  *is minimum*

167 We look for a minimum subset  $V'$  of  $V$  containing  $S$ , such that the subgraphs of all  
 168 snapshots induced by  $V'$  are connected (subgraph of  $G$  induced by  $X$  : subgraph containing  
 169  $X$  and all edges linking vertices of  $X$ , denoted  $G[X]$ ).

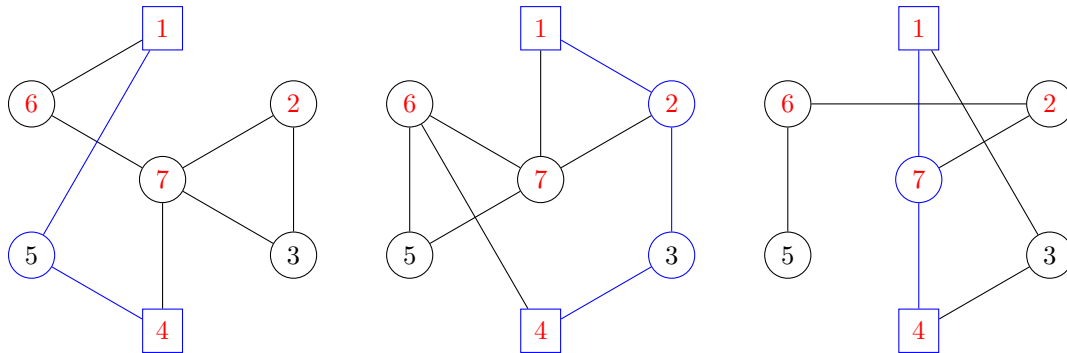
170 The subset  $V'$ , still called the Steiner set, is a subset of nodes that keep the terminals  
 171 connected. But in fact the condition verified by each  $G'_i$  is stronger: it keeps the connectivity  
 172 between all vertices of  $V'$ , not only the terminals.

#### 173 **2.2.2 Remarks**

174 Several elements are important to notice regarding the problem given by Definition 1.

- 175 **1.** When  $T = 1$ , this problem is the Steiner Tree problem.

- 176 2. Enforcing the subset  $E'_i$  to be the same at each time step makes this problem solvable as  
 177 a static problem by intersecting all  $G_i$
- 178 3. The vertices from the optimal Steiner tree of  $S$  at a given time step do not necessarily  
 179 belong to the optimal solution.
- 180 4. The union of the optimal Steiner trees at each time step is not optimal.
- 181 5. The intersection of the Steiner trees obtained by the solution of this problem at each  
 182 time step is not a connected graph.
- 183 6. Choosing  $V' = V$  is always a solution, but the worst possible in terms of cardinality.



■ **Figure 1** A dynamic graph with 7 vertices and 3 time steps. The graph is represented at each time step.

### 184 2.2.3 Example

185 Figure 1 presents an example of the evolution of a dynamic graph on 3 time steps.

186 Rectangular vertices (vertices 1 and 4) correspond to the terminal subset  $S$ . Vertices with  
 187 identification in red (vertices 1, 2, 4, 6 and 7) correspond to the subset  $V'$  of the optimal  
 188 solution of the Fully Connected Steiner Set Problem (Definition 1).

189 Blue circled vertices and blue edges are the optimal Steiner trees for the subset  $S$  at each  
 190 time step. Their size are 3, 4 and 3, respectively. The set  $\{1, 2, 3, 4, 5, 7\}$  resulting from their  
 191 union is not connected for the thierd snapshot.

192 The Steiner set for this example is  $V' = \{1, 2, 4, 6, 7\}$ , its cardinality is 5.

193 This example illustrates the remarks presented in section 2.2.2. The blue vertex set  
 194 and the red vertex set are not equivalent, which is the point of remark 3. Intersecting the  
 195 Steiner trees at each time step (in other words, intersecting blue edges and vertices) gives an  
 196 independent set in this example. Therefore, no solution can be extracted from this, which is  
 197 the point of remark 5. The union of the optimal Steiner Sets at each time step (remark 4) is  
 198 not always connected in this specific example, and as stated by remark 6, taking all vertices  
 199 gives of course a solution of value 7 instead of 5.

### 200 2.2.4 Issues

201 This *Fully Connected Steiner Set* problem cannot be simply reduced to the computation of a  
 202 Steiner tree on some static graph built from the dynamic graph. Therefore, it deserves a  
 203 specific study.

204 Unfortunately, the full connectivity constraint on each  $G'_i$  is rather artificial with regard to  
 205 applications. Remember the initial goal of building Steiner sets is to keep only the terminals

206 connected. One could wonder why enforce the connectivity of the whole Steiner Set as long  
 207 as the terminals are connected. An optimal solution of this problem might keep some edges  
 208 and vertices that will not be used to connect the terminals at each snapshot, *thus significantly*  
 209 *increasing the overall cost* (this remains true if costs are added to edges). This is the case in  
 210 the example presented before. For instance, 4 edges are necessary to ensure connectivity at  
 211 each snapshot. But during the first and third time steps, only two edges are necessary to  
 212 connect the two terminals.

213 Another issue is the difficulty to solve the problem in practice. Note however that the  
 214 problem is polynomial when the size of the searched Steiner set is bounded. Indeed, for a  
 215 fixed maximum size of the Steiner set, there is a polynomial number of sets to test (if we  
 216 consider the terminals to be part of the Steiner Set, the number of terminals does not matter).  
 217 And this test consists in searching for the  $L$  minimum spanning trees associated with the  
 218 induced subgraphs. First we should enumerate all connected subsets  $W$  of  $V$  containing  $S$   
 219 and find the minimum spanning tree of the graph induced by  $W$  at each time step. The  
 220 optimal solution is the subset  $W$  for which the sum of the weights at each time step is  
 221 minimum. This method is obviously not applicable in practice as their might be as much as  
 222  $2^{n-k}$  sets  $W$  to check on a graph with  $n$  vertices and with  $S$  having  $k$  vertices.

## 2.3 Third model: The terminal set is connected, not the Steiner set

### 2.3.1 Definition

225 From the previous extension of the Steiner Problem, one major remark was that it seemed  
 226 artificial to keep the whole Steiner set connected because the objective is to connect the  
 227 terminals. This is how we came to another possible definition to extend the Steiner problem  
 228 to dynamic graphs: the Partially Connected Steiner Set.

229 ► **Definition 2** (Partially Connected Minimum Steiner Set). *Let us consider some dynamic*  
 230 *graph  $\mathcal{G}$ . For a given vertex set  $S \subset V$  of terminals, find  $V'$  with  $S \subset V' \subset V$ , and*  
 231  *$E'_i \subset E_i \forall i \leq L$  such that:*

- 232 ■ *All vertices of  $S$  are part of the same connected component in the static graph  $G'_i = (V', E'_i)$ ,*  
 233  *$\forall i \leq L$ .*
- 234 ■  *$|V'|$  is minimum*

235 The (partially connected) Steiner set  $V'$  is the subset of nodes that keep the terminals  
 236 connected for the whole time interval of study of the graph, at a minimum cost.  $V'$  itself is  
 237 not necessarily connected. This is an answer to a drawback of the previous definition.

### 2.3.2 Remarks and complexity issues

239 Most remarks from the previous definition remain valid. Again, when  $T = 1$ , this problem is  
 240 the Steiner Tree problem. We still cannot deduce an optimal solution from the solution of  
 241 static Steiner Problem at each time step. *However, the problem does not directly extend to*  
 242 *the case of weights on the edges. Indeed, if one minimizes the sum of weights of the Steiner*  
 243 *trees chosen at each snapshot, then the optimal solution is simply to take  $V' = V$  and the*  
 244 *optimal Steiner trees at each snapshot. It is then equivalent to the direct extension given in*  
 245 *Section 2.1.*

246 Clearly, there is a compromise to accept between the size of the Steiner set and the overall  
 247 cost of the chosen edges. Anyway, it seems natural to first compute the minimum size of a  
 248 Steiner tree, and then to compute the solutions with minimum cost for a *fixed* value of the  
 249 cardinality of  $V'$ .



250 Finding the (partially connected) Dynamic Minimum Steiner Set, denoted *DMSS*, is  
 251 NP-hard in the static case, even for bipartite[12] or for chordal graphs[21]. On the other  
 252 hand, one might consider the case of a small number of terminals. Even with arbitrary costs,  
 253 the problem with two terminals is easy in the static case as the Steiner tree reduces to a  
 254 single path.

255 In figure 1, there are two terminals. It is easy to see that  $\{6, 7\} \cup \{1, 4\}$  is the Steiner set  
 256 of minimum cardinality 4. Remember for the fully connected Steiner set variant, the optimal  
 257 value is 5 for the same example. Note that it is easy to build examples for which the optimal  
 258 cost for the second model is arbitrary larger than the optimal cost for the third model. In  
 259 the next section, we consider the case of the Partially Connected Minimum Steiner tree, with  
 260 two terminals, and we show that this problem is NP-hard. It is noted *DMSS2* for (Partially  
 261 Connected) Dynamic Minimum Steiner Set with 2 terminals.

## 262 **3 Partially Connected Dynamic Minimum Steiner Set with 2 Terminals**

### 263 **3.1 Definition**

264 ► **Definition 3** (Partially Connected Minimum Steiner Set with 2 Terminals). *Let us consider*  
 265 *some dynamic graph  $\mathcal{G}$ . For a given vertex set  $S \subset V$  of terminals, with  $|S| = 2$ , find  $V'$  with*  
 266  *$S \subset V' \subset V$ , and  $E'_i \subset E_i \forall i \leq L$  such that:*

- 267 ■ *All vertices of  $S$  are part of the same connected component in  $G'_i = (V', E'_i)$*
- 268 ■ *the cardinality of  $V'$  is minimum*

### 269 **3.2 Remarks**

270 This problem is very easy when  $T = 1$  as this is a shortest path problem in a graph where  
 271 all edges have weight 1. But it is not so easy in the dynamic context, and some extreme  
 272 examples can be given.

273 Figure 2 presents a dynamic graph on 4 time steps. The terminal vertices are the squared  
 274 ones. They are directly connected by an edge at each time step, therefore no extra vertex is  
 275 necessary to connect them.

276 Figure 3 presents a dynamic graph on  $n$  time steps. In each  $G_i$ , there is exactly one  
 277 path of length 2 between the two terminals going through a different vertex at each time  
 278 step. In this case, every vertex of the graph must be in the Steiner Set to keep the terminals  
 279 connected.

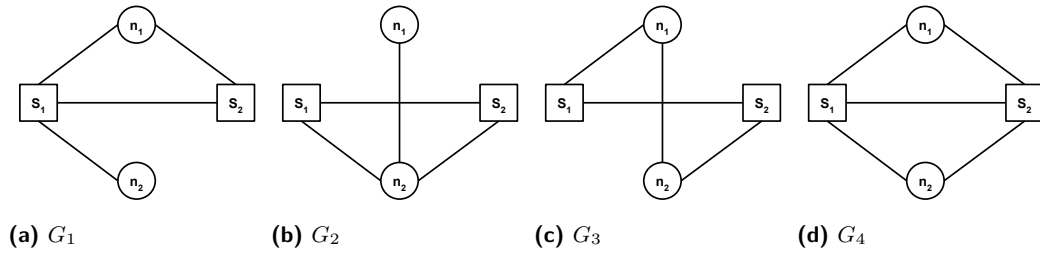
280 Figure 4 presents a dynamic graph on 4 time steps. At each time step, the terminals are  
 281 connected by two distinct paths: one of length 3 and one of length at most 2. If we consider  
 282 the vertices on the shortest path at each time step for the Steiner Set, we need 3 vertices to  
 283 keep the terminals connected. But if we connect the terminals at each time step by a longer  
 284 path (one of length 3), we only need 2 extra vertices for the Steiner set. This example also  
 285 shows that the shortest path at each time step does not help getting an optimal solution, and  
 286 that intersecting the graphs  $G_i$  does not help either. Indeed, the path of length 3 present at  
 287 each time step goes through the same two vertices but not the same edges.

288 Even though this problem is easy to describe, it is difficult to solve. This is what we  
 289 prove in Section 3.3.

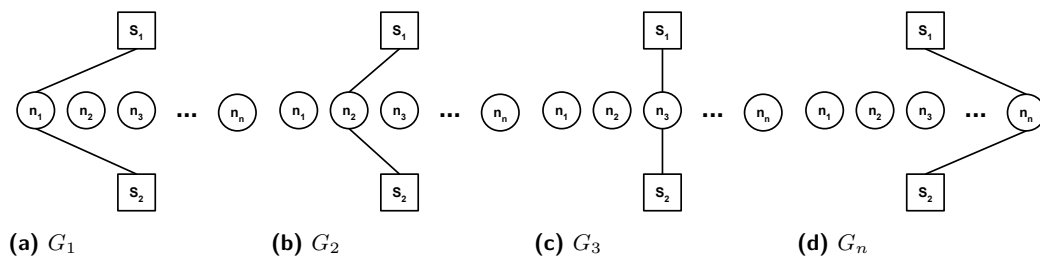
### 290 **3.3 NP-Completeness for DMSS2 decision problem**

291 Let us consider the associated decision problem. [Note that for simplicity, we introduce the](#)  
 292 [set  \$V\_s\$  as the considered Steiner set minus the two terminals, denoted by  \$a\$  and  \$b\$ .](#)

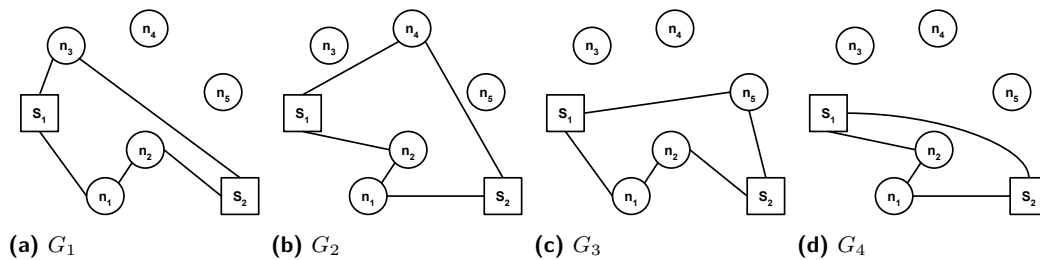




■ **Figure 2** Dynamic graph on 4 time steps. The terminal vertices are the squared vertices.



■ **Figure 3** Dynamic graph on  $n$  time steps. The terminal vertices are the squared vertices.



■ **Figure 4** Dynamic graph on 4 time steps. The terminal vertices are the squared vertices.

293 ► **Definition 4** (*DMSS2 decision problem*). Let  $\mathcal{G} = (G_i)_{1 \leq i \leq L}$  be a dynamic graph such that:

294 ■  $G_i = (V, E_i)$ ,  $i \leq L$

295 ■  $\cup_{i \leq L} E_i = E$

296 Let  $a \in V$  and  $b \in V$  be two vertices called terminals. Let  $V_s \subset V - \{a, b\}$  be a vertex set  
297 such that:

298 ■  $\forall i \leq L$  there is a path between  $a$  and  $b$  in  $G_i$  using only vertices from  $V_s$

299 **Question:** for a given non-negative integer  $p$ , is there a vertex set  $V_s$  of size  $p$ ?

300 The NP-completeness of *DMSS2* decision problem is proven with a reduction from the  
301 vertex cover problem (VC). This problem is defined as follows.

302 ► **Definition 5** (VC). Let  $G = (V, E)$  be a graph, and let  $V_c \subset V$  be a vertex set such that  
303  $\forall (u, v) \in E$ ,  $u \in V_c$  or  $v \in V_c$ .

304 **Question:** for a given non-negative integer  $k$ , is there a vertex set (a cover set)  $V_c$  of  
305 size  $k$ ?

306 In order to prove that the *DMSS2* decision problem is NP-complete, we first need to show  
307 that there is a polynomial reduction from a Vertex Cover instance to a *DMSS2* instance  
308 (Lemma 6). Then we need to show that a valid VC instance gives a valid *DMSS2* instance  
309 (Lemma 7) and that a valid *DMSS2* instance gives a valid VC instance (Lemma 8) using the  
310 polynomial reduction. An instance of a given decision problem is said valid if the answer to  
311 the question is YES for this instance.

312 ► **Lemma 6.** *There is a polynomial reduction from VC to DMSS2.*

313 **Proof.** Let us build a *DMSS2* instance on a dynamic graph  $G^{DYN}$  from a VC instance on  
314 graph  $G = (V, E)$ .

315 ■ For each vertex  $u$  in  $G$ , there is a vertex  $u$  in the dynamic graph  $G^{DYN}$ ;

316 ■  $G^{DYN}$  has two extra vertices  $a$  and  $b$ ;

317 ■ For each edge  $e = (u, v)$  in  $G$ , there is a corresponding time step  $i_e$  in  $G^{DYN}$  and  $G_{i_e}^{DYN}$   
318 has 4 edges:  $(a, u)$ ,  $(u, b)$ ,  $(a, v)$ ,  $(v, b)$ .

319 Vertices  $a$  and  $b$  in  $G^{DYN}$  are the terminal vertices. Figure 5 displays an example of such  
320 transformation.

321 If graph  $G$  has  $n$  vertices and  $m$  edges,  $G^{DYN}$  has  $m$  time steps,  $n + 2$  vertices, and each  
322  $G_i^{DYN}$  has 4 edges.

323 The *DMSS2* instance is built in polynomial time ( $O(m + n)$ ) from the VC instance. ◀

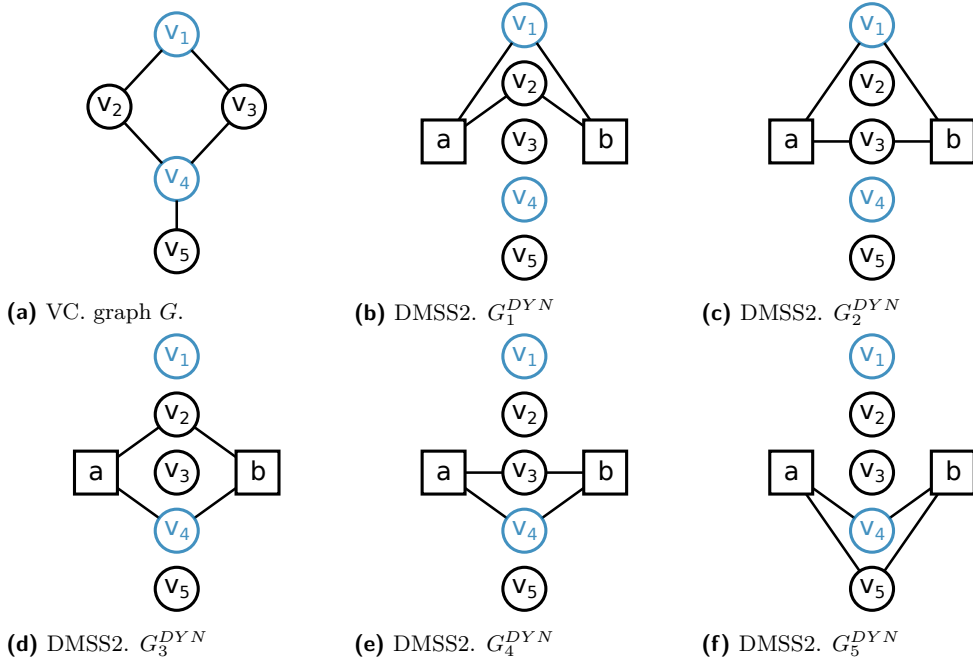
324 ► **Lemma 7.** *If a VC instance is valid, then the corresponding DMSS2 instance is also valid.*

325 **Proof.** Consider a vertex cover  $V_c$  of size  $p$ , and let  $V_s$  be the vertex set of  $G^{DYN}$  with the  
326 same vertices. Therefore  $V_s$  also has size  $p$ .

327 Let us consider some time step  $i$ . The corresponding graph  $G_i^{DYN}$  contains exactly two  
328 paths of length 2 between  $a$  and  $b$  using some vertex  $u$  and some vertex  $v$ . By construction  
329 of  $G^{DYN}$ ,  $(u, v)$  is an edge of  $G$ . Hence either  $u \in V_c$  or  $v \in V_c$  (or both). Therefore, in  
330  $G^{DYN}$  either  $u \in V_s$  or  $v \in V_s$  (or both). As at least one of them is in  $V_s$ , there exists in  
331  $G_i^{DYN}$  a path between  $a$  and  $b$  using only vertices from  $V_s$ .

332 This is true for each time step in  $G^{DYN}$ , so  $V_s$  is a solution of the *DMSS2* instance. ◀

333 ► **Lemma 8.** *If a DMSS2 instance is valid, then the corresponding VC instance is also valid.*



■ **Figure 5** Transforming a VC instance with 5 edges to a  $DMSS2$  instance with 5 time steps. The sets  $V_C = V_S = \{v_1, v_4\}$  are in blue.

334 **Proof.** Consider the set  $V_s$  of  $G^{DYN}$  of size  $p$ , and let  $V_c$  be the vertex set of  $G$  with the  
335 same vertices. Therefore  $V_c$  also has size  $p$ .

336 Let  $e = (u, v)$  be an edge of  $G$ . For the corresponding time step  $i_e$ ,  $G_{i_e}^{DYN}$  has two paths  
337 of length exactly 2 between  $a$  and  $b$ , with  $u$  and  $v$  the two vertices on those paths. As  $V_s$  is  
338 a solution of the  $DMSS2$  problem, it means that either  $u$  or  $v$  (or both) are in  $V_s$ . Therefore,  
339 as  $V_c$  and  $V_s$  contain the same vertices,  $u$  or  $v$  (or both) are in  $V_c$ .

340 This is true for all edges of  $G$ , hence  $V_c$  is a solution of the Vertex Cover problem. ◀

341 Figure 5 shows an example where both instances are valid for  $p = 2$ . The associated sets  
342  $V_c$  and  $V_s$  are in blue. In Figure 5a, the blue vertices are a vertex cover. Indeed, each edge  
343 of the graph is adjacent to at least one blue vertex. In Figures 5b to 5f, the blue vertices,  
344 plus the terminal vertices, form a Steiner Set. Indeed, at each time step of the graph, vertex  
345  $a$  and vertex  $b$  are connected by a path going only through blue vertices.

346 ► **Theorem 9.** *The decision problem  $DMSS2$  is NP-complete.*

347 **Proof.** Clearly,  $DMSS2$  is in  $NP$  as a proof of *YES* answer is obtained by providing one  
348 Steiner set and one tree per snapshot containing all terminal vertices and some other vertices  
349 from the Steiner set.

350 The remaining of the proof follows directly from lemmas 6, 7 and 8. ◀

351 Remember that 3 variants were presented in the previous section; we can remark that in  
352 the first variant Steiner sets are independently computed for each snapshot. Hence with 2  
353 terminals it can be done through  $L$  breadth first searches, and the problem is polynomial.  
354 In the second variant, the whole Steiner set (including the two terminals) must be connected  
355 at each snapshot. The same proof than for partially connected  $DMSS$  can be used. Indeed,  
356 it suffices to modify slightly the reduction (see lemma 6) as follows: in  $G_i^{DYN}$  all vertices

357 but  $a$  and  $b$  form a clique. Thus, the solution corresponding to the vertex cover set remains  
 358 the same as it still needs to contain at least one extremity of each edge of  $G$  to connect  $a$   
 359 and  $b$ . Because of the clique edges, the restriction of  $G_i^{DYN}$  to the solution set is connected  
 360 for all  $i$ , so this vertex set is indeed a solution of the fully connected *DMSS*. So this variant  
 361 is also NP-complete even with two terminals.

### 362 3.4 NP-Completeness for two time-steps

363 The dynamic Steiner optimization problem with two terminals is NP-hard. But in the  
 364 reduction, the number of time-steps of the resulting dynamic graph is  $m$ , so at least  $\Theta(n)$ .  
 365 What happens when the life-time is strictly smaller? In particular, the problem should be  
 366 easier when the life-time is bounded by a constant. We now prove that the problem is still  
 367 NP-hard, even when the lifetime is 2.

368 ► **Definition 10** (*DMSS2x2 decision problem*). Let  $\mathcal{G} = (G_1, G_2)$  be a dynamic graph with  
 369  $G_1 = (V, E_1)$ ,  $G_2 = (V, E_2)$ . Let  $a \in V$  and  $b \in V$  be two vertices called terminals. Let  
 370  $V_s \subset V \setminus \{a, b\}$  be a vertex set such that:

371 ■  $\forall i = 1, 2$  there is a path between  $a$  and  $b$  in  $G_i$  using only vertices from  $V_s$

372 **Question:** for a given non-negative integer  $p$ , is there a vertex set  $V_s$  of size  $p$ ?

373 The NP-completeness of *DMSS2x2* decision problem is proven with a reduction from  
 374 the SAT problem. We first show that there is a polynomial reduction from a SAT instance  
 375 to a *DMSS2x2* instance (Lemma 11). Then we show that a valid SAT instance gives a  
 376 valid *DMSS2x2* instance (Lemma 12) and that a valid *DMSS2x2* instance gives a valid SAT  
 377 instance (Lemma 13) using the polynomial reduction.

378 ► **Lemma 11.** *There is a polynomial reduction from SAT to *DMSS2x2*.*

379 **Proof.** Let us build a *DMSS2x2* instance on a dynamic graph  $G^{DYN} = (G_1 = (V, E_1), G_2 =$   
 380  $(V, E_2))$  from a SAT instance. The SAT instance formula is given as a conjunctive normal  
 381 form, with  $n$  variables and  $c$  clauses. Graph  $G_1$  is first built as a level graph, one level per  
 382 variable and two vertices by level. Then each initial vertex is replaced by a chain of  $c$  vertices.  
 383 Some additional vertices are added in a straightforward way. Graph  $G_2$  is also built as a  
 384 level graph, one level per clause and  $c$  vertices by level. Vertices  $c_j$  are then used to separate  
 385 these levels. The remaining vertices are connected to  $a$  to insure connexity. Thus, a shortest  
 386 path in  $G_1$  is associated to an interpretation of the formula (i.e. a boolean assignment for  
 387 each variable), and a shortest path of  $G_2$  to one literal per clause. More formally:

388 ■ for each vertex variable  $x_i$  and clause  $C_j$ , there are two vertices  $v_i^j$  and  $\bar{v}_i^j$  in  $V$ ;

389 ■ for each clause  $C_j$ ; except  $C_n$  ( $j = 1 \rightarrow (n - 1)$ ), there is a vertex  $c_j$  in  $V$ ;

390 ■  $G^{DYN}$  has two extra vertices  $a$  and  $b$ , also denoted  $c_0$  and  $c_c$ ;

391 ■  $E_1$  contains the following edges:

392  $(a, v_1^1); (a, \bar{v}_1^1); \forall i = 1 \rightarrow n, \forall j = 1 \rightarrow c - 1, (v_i^j, v_i^{j+1}); (\bar{v}_i^j, \bar{v}_i^{j+1});$

393  $\forall i = 1 \rightarrow n - 1, (v_i^c, v_{i+1}^1); (v_i^c, \bar{v}_{i+1}^1); (\bar{v}_i^c, v_{i+1}^1); (\bar{v}_i^c, \bar{v}_{i+1}^1);$

394  $(v_n^c, c_1); (\bar{v}_n^c, c_1); \forall j = 1 \rightarrow c - 1, (c_j, c_{j+1}).$

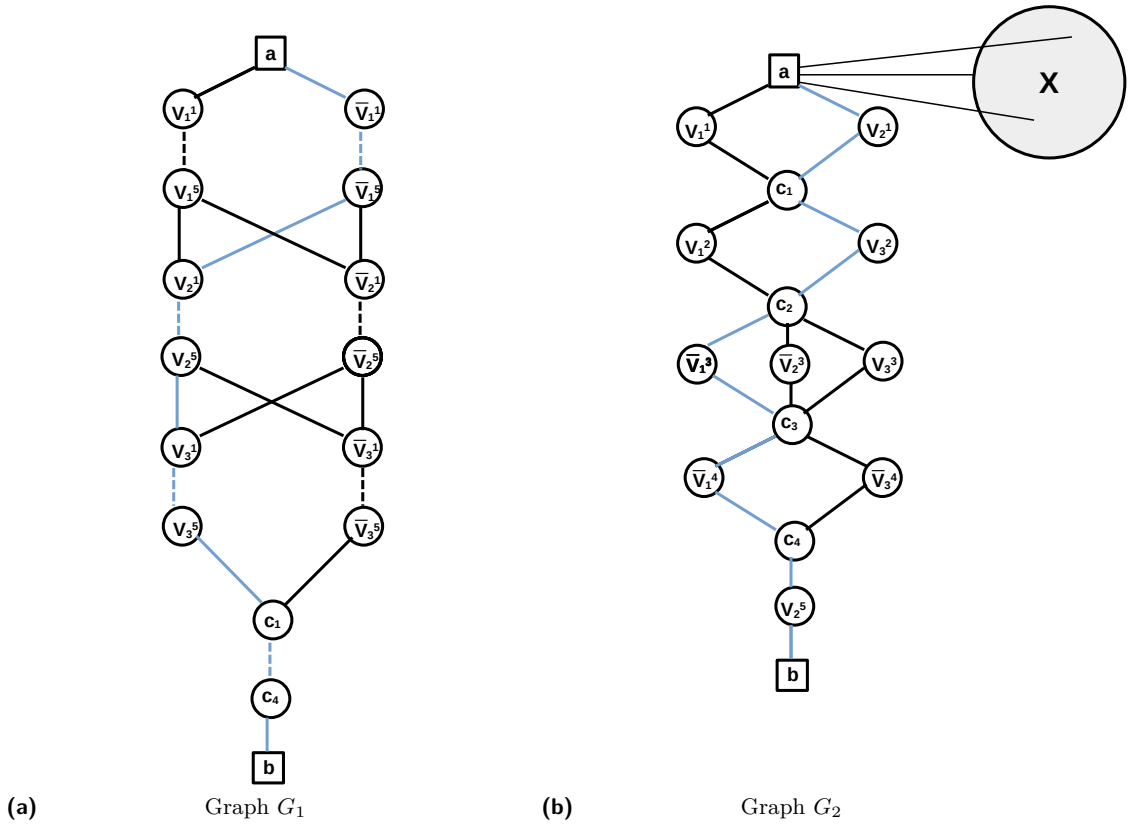
395 ■  $E_2$  contains the following edges:

396  $\forall j = 1 \rightarrow c, \forall x_i \in C_j, (c_{j-1}, v_i^j); (v_i^j, c_j);$

397  $\forall j = 1 \rightarrow c, \forall \bar{x}_i \in C_j, (c_{j-1}, \bar{v}_i^j); (\bar{v}_i^j, c_j);$

398  $\forall j = 1 \rightarrow c, \forall x_i, \bar{x}_i \notin C_j, (a, v_i^j); (a, \bar{v}_i^j).$

399 Vertices  $a$  and  $b$  in  $G^{DYN}$  are the terminal vertices. Figure 6 displays an example of such  
 400 reduction.  $G^{DYN}$  has 2 time-steps,  $2nc + c + 1$  vertices.  $E_1$  has  $(2n + 1)(c - 1) + 4n$  edges;



■ **Figure 6** Example of reduction from instance 1 of SAT. Dashed lines represent a path with some inner vertices.  $X$  is the set of all vertices not present elsewhere in  $G_2$ , all linked to  $a$ . In blue, both paths of a valid solution.

401  $E_2$  has  $2cn + \sum_j s_j \leq 3cn$  ( $s_j$  is the number of literals in clause  $C_j$ ). Finally, the question  
 402 for DMSS2x2 is: is it a vertex set  $V_s$  of size  $p = nc + (c - 1)$ ?

403 The DMSS2x2 instance is built in polynomial time  $O(nc)$  from the SAT instance. ◀

404 Figure 6 shows a simple example of reduction. The SAT instance has 3 variables and 5  
 405 clauses. Its formula is:

$$406 (x_1 + x_2)(x_1 + x_3)(\bar{x}_1 + \bar{x}_2 + x_3)(\bar{x}_1 + \bar{x}_3)x_2. \quad (1)$$

407 The obtained instance of DMSS2x2 has 36 vertices. Both instances are valid. From the paths  
 408  $P_1$  and  $P_2$  of a solution  $V_s$  for DMSS2x2, one can easily get the corresponding solution for  
 409 SAT:  $x_1$  is false,  $x_2$  and  $x_3$  are true.

410 ▶ **Lemma 12.** *If a SAT instance is valid, then the corresponding DMSS2x2 instance is also*  
 411 *valid.*

412 **Proof.** Consider a solution of the SAT instance, that is an interpretation  $I$  that satisfies the  
 413 formula. Let us consider the path  $P_1$  from  $a$  to  $b$  in  $G_1$  so that: if  $x_i \in I$ , then  $v_i^j$  belongs  
 414 to  $P_1$  for all  $j$ ; and if  $\bar{x}_i \in I$ , then  $\bar{v}_i^j$  belongs to  $P_1$  for all  $j$ . The number of vertices of  $P_1$   
 415 ( $a$  and  $b$  excluded) is exactly  $nc + c - 1$  (and there are exactly  $2^n$  shortest paths). Let us  
 416 now consider  $G_2$ . As  $I$  is a solution of the SAT instance, it contains at least one literal per  
 417 clause  $C_j$ , say  $\tilde{x}_{i_j}$ . So in  $G_2$ , one can build an elementary path  $P_2$  from  $a$  to  $b$  including the

418 corresponding vertex of this literal ( $v_i^j$  or  $\bar{v}_i^j$ ) for all  $C_j$ , plus the  $c_j$ . By construction, all  
 419 vertices of  $P_2$  belong to  $P_1$ . So we have built a feasible solution for DMSS2x2 of size exactly  
 420  $p = nc + c - 1$ . ◀

421 ▶ **Lemma 13.** *If a DMSS2x2 instance is valid, then the corresponding SAT instance is also*  
 422 *valid.*

423 **Proof.** From the construction of  $G^{DYN}$ , we know that all paths of  $G_1$  contain at least  
 424  $nc + c - 1$  vertices (as usual,  $a$  and  $b$  excluded), and all elementary paths of  $G_2$  contain  
 425 exactly  $2c - 1$  vertices. Let us consider a set  $V_s$  of  $G^{DYN}$  of size  $p = nc + c - 1$ , supposed to  
 426 be a valid solution, and denote  $P_1$  and  $P_2$  the corresponding paths. As  $V_s$  is a valid solution,  
 427  $P_1$  contains exactly  $nc + c - 1$  vertices, and all vertices of  $P_2$  must be included into  $P_1$ .

428 Let us build an interpretation  $I$  of the SAT instance as follows: if some  $v_i^j$  belongs to  
 429  $P_2$ , then  $x_i$  belongs to  $I$ . If some  $\bar{v}_i^j$  belongs to  $P_2$ , then  $\bar{x}_i$  belongs to  $I$ . Note that it is  
 430 impossible to have both  $v_i^j$  and  $\bar{v}_i^j$  in  $V_s$  for some  $i$ , as they belong to  $P_1$  and by construction,  
 431 for a given  $i$  either all  $v_i^j$  or all  $\bar{v}_i^j$  belong to  $P_1$ . By construction of  $G_2$ , the path  $P_2$  contains  
 432 at least one vertex associated to one literal of each clause. However, it is possible that for a  
 433 given  $i$  it contains no vertex of the form  $v_i^j$ , and no vertex of the form  $\bar{v}_i^j$ . In that case, one  
 434 may add to  $I$  either  $x_i$  or  $\bar{x}_i$ , indifferently. That way, interpretation  $I$  satisfies all clauses, so  
 435 it is a valid SAT instance. ◀

437 ▶ **Theorem 14.** *The decision problem DMSS2x2 is NP-complete.*

438 **Proof.** Clearly, DMSS2x2 is in NP as DMSS2 is.  
 439 The remaining of the proof follows directly from lemmas 11, 12 and 13. ◀

440 Note that in the proposed reduction, both snapshots  $G_1$  and  $G_2$  are connected. Further-  
 441 more, one may observe that all vertices of  $V_s \cup \{a, b\}$ , if  $V_s$  is a valid solution of DMSS2x2,  
 442 are connected. This is true in particular for the vertices of  $X$  that are all directly connected  
 443 to  $a$  in  $G_2$ . This choice was not mandatory to prove theorem 14, but was done to prove  
 444 easily the following result.

445 ▶ **Theorem 15.** *The decision problem associated to variant two: Fully Connected Minimum*  
 446 *Steiner Set, with 2 terminals and 2 snapshots, is NP-complete.*

447 **Proof.** The proof is immediate. Indeed, in the second variant there is an additional condition  
 448 on the Steiner set: its vertices must be connected through other vertices of the Steiner set.  
 449 This property is verified for the reduction used in the above proof: in  $G_1$ , the chosen vertices  
 450 form a path. In  $G_2$ , some of the chosen vertices form a path from  $a$  to  $b$ , and the others are  
 451 directly connected to  $a$ . Therefore the proof directly applies to this variant. ◀

## 4 Algorithmic issues

453 This section focuses on efficiently solving the optimization problem associated with DMSS:  
 454 costs on the edges are not considered, only the cardinality of the Steiner set matters. We first  
 455 consider the problem of finding a Steiner set of given cardinality. Throughout this section,  
 456  $s$  is the number of terminals,  $k$  is the cardinality of the Steiner sets, and  $k' = k - s$  is the  
 457 number of vertices to add to the terminal vertices.

458 Our objective is to design efficient (polynomial) algorithms when  $k$  is fixed.

## 4.1 Basic ideas

For a given time step  $i$ , there might be a very large number of vertex sets of cardinality  $k'$  that allow the terminals to be connected. These sets will be called *candidates* for time step  $i$ . An obvious upper bound of this number is the number of vertex subsets of cardinality  $k'$  among  $V - S$ , that is  $\binom{n-s}{k'}$ . The (dynamic) Steiner sets we are looking for belong to this set but, hopefully, are much less numerous. Still, for dynamic graphs with a high number of edges in the underlying graph and a slow dynamic, this bound might be close. Note however that the problem on such graphs should not be very interesting from the applications point of view. Furthermore, a Steiner set of the given cardinality will be found with high probability by a simple, and fast, breadth first search on the intersection graph (intersection of all snapshots). We shall discuss in the next section on these questions relative to the instance types.

A Steiner set of cardinality  $k$  must by definition connect the terminals for each time step. Therefore, the algorithmic possibilities are straightforward. Let us discard the naive idea that consists in computing independently all candidate subsets for all time steps, and then computing their intersection. It is much more efficient to use an iterative process: suppose we have a set of candidates  $PSS$  for time steps  $1, \dots, i$ . At iteration  $i + 1$ , only the solutions in  $PSS$  which are also candidates for time step  $i + 1$  are kept. The process starts with all candidates for  $i = 1$ . It ends when  $i = L$  with all Steiner sets of cardinality  $k$ . This principle is used in the algorithm below. In the remaining of the section, we shall precise the different steps and give some complexity issues.

■ **Algorithm 1** FindSteinerFixedSize( $\mathcal{G}, k$ )

---

Input:  $\mathcal{G} = (G_i)_{i \in \{1, \dots, T\}} = (V, E_i)$ ;  $k \leq |V|$   
 Output: all Steiner sets of cardinality  $k$ .

$PSS \leftarrow \text{FindCandidates}(G_1, k)$   
 for  $i$  from 2 to  $L$  do  
 |  $PSS \leftarrow \text{Update}(PSS, i)$ ;  
 | If  $(PSS = \emptyset)$  then STOP.  
 end  
 Return:  $PSS$

---

## 4.2 Algorithm details and complexity

Method  $\text{FindCandidates}(G_1, k)$  computes all candidates for  $G_1$ . Unfortunately, there is no other way than enumerating all subsets of  $V - S$  of cardinality  $k'$ . For each subset  $K$ , determining whether it is a candidate can be done in two steps. First, build the subgraph of  $G_1$  generated by  $S \cup K$ . Second, perform some search from any vertex of  $S$  on this subgraph. Subset  $K$  is candidate if and only if all other vertices of  $S$  are found during this search.

Method  $\text{Update}(PSS, i)$  uses the same procedure as  $\text{FindCandidates}$ , but only on the elements of  $PSS$ , and on  $G_i$ .

So the core procedure of the algorithm is a search on a small subgraph. Its complexity is  $\mathcal{O}(m')$  where  $m'$  is the number of edges of the subgraph. This number is bounded by  $k^2$  and by  $m$ , the number of edges of the underlying graph.

In the worst case (see above), the number of candidates remains very high throughout the iteration process, it is majored by  $\binom{n-s}{k'}$ , hence by  $\frac{(n-s)^{k'}}{k'!}$ .



494 ► **Theorem 16.** *The worst case complexity of algorithm 1 as a function of the number*  
 495 *of time steps  $L$ , the number of vertices  $n$ , the number of terminals  $s$ , and the size of the*  
 496 *computed Steiner sets  $k$ , is  $\Theta(T \times (n - s)^{k-s} \times \frac{k^2}{(k-s)!})$ . When  $k$  and  $s$  are fixed parameters,*  
 497 *the complexity is  $\Theta(T \times n^{k-s})$ .*

498 **Proof.** The first part is immediate from the number of iterations, the maximum number of  
 499 candidates at each iteration, and the complexity of the search for each candidate.

500 The best case is obtained when no Steiner set exists for  $G_1$ . In that case, the algorithm tests  
 501 all candidates for  $G_1$  then stops. ◀

502 These recent years, many studies focused on revisiting classical problems in the light of  
 503 parameterized complexity, and this is of course true for Steiner problems. The main results  
 504 for Steiner, with a presentation of the basics of parameterized complexity, are summarized in  
 505 [14]. When the parameter is the number of terminals  $s$ , the minimum cardinality problem  
 506 is *FPT* (Fixed Parameter Tractable). But it is  $W[2]$ -hard if the parameter is the size of  
 507 the Steiner set (the problem is still *FPT* if the graph is planar). This means no algorithm  
 508 solves the problem in time  $f(k) \cdot |I|^{\mathcal{O}(1)}$ , where  $|I|$  is the instance size and  $f$  is an arbitrary  
 509 function depending only of  $k$ , and this of course remains true in the dynamic case. [Theorem](#)  
 510 [16](#) however shows that *DMSS* is in the *XP* class when  $k$  (or  $k' = k - s$ ) is considered, but  
 511 leaves open whether it is *FPT* for parameter  $s$ .

### 512 4.3 Some additional remarks on the algorithms

513 The worst case complexity is the same as that of the naive method that computes all  
 514 candidates at each iteration. It is expected that the average and experimental complexity  
 515 will be much better.

516 If the snapshot graphs are sparse, it is expected that the number of kept candidates  
 517 diminishes very fast as the number of iterations increases, and eventually equals 0. Then the  
 518 algorithm stops and returns the empty set. One expects also for these instances that most  
 519 of the work is done during first step. Finally, Note that a preprocessing phase accelerates  
 520 greatly the tests (see next section): a graph search checks if a Steiner tree exists in the  
 521 underlying graph. If not, the result is of course the empty set.

522 Algorithm 1 returns all Steiner sets of size  $k$ . To return Steiner sets of minimal size  $k^*$ ,  
 523 two approaches are possible. The direct one is to use a dichotomy search and repeatedly  
 524 call the algorithm. The number of calls is then  $\mathcal{O}(\log \kappa)$  where  $\kappa$  is an upper bound of  
 525  $k'^* = k^* - s$  (at most  $n - s$ ). The second approach consists in keeping at each step the  
 526 candidates that are minimal for the inclusion (that is, removing any vertex removes the  
 527 subset from the candidate set). Updating this set at each time step is more complicated,  
 528 and it is not clear whether this approach is faster.

529 Finally, let us consider the problem of Steiner sets of fixed size and minimal cost. Due to  
 530 the nature of the problem, a last step must be added to algorithm 1: for each final Steiner set,  
 531 compute its weight. This implies to compute  $L$  minimum spanning trees (on a graph with  $k$   
 532 vertices) for each Steiner set, which costs at most  $\mathcal{O}(T \times k^2 \times \log k)$  with Prim or Kruskal  
 533 algorithms. Note that this complexity is polynomial in all parameters, and independent of  $n$   
 534 (but of course the number of Steiner sets is not independent of  $n$ , and not always polynomial).

## 535 5 Experimental study

536 In this section, we present an experimental study of our algorithm, both on randomly  
 537 generated graphs and on real-world networks.

538 **5.1 Generated graphs**

539 The experimental study on randomly generated graphs was performed using the GraphStream  
 540 Java library<sup>2</sup>[9]. The virtual machines used for this experiment have 8 core processors with  
 541 64 GB of RAM.

542 **5.1.1 Experimental settings**

543 In order to run experiments on our algorithm, we generated dynamic graphs using a method  
 544 previously presented in [20] to test connected component computation. First we generate  
 545 the underlying graph, then we add dynamicity to the edges using a Markovian process.

546 The underlying graph is generated using generators from the GraphStream library. We  
 547 test three different types of graphs that present specific features:

- 548 1. Random graphs, corresponding to the Erdős-Rényi model[10]. It is the most common  
 549 way of randomly generating a static graph. GraphStream Random Graph generator is  
 550 used. This generator adds a vertex and randomly connects it to the other vertices of the  
 551 graph. This operation is repeated for each vertex added.
- 552 2. Regular graphs, which are generated using GraphStream Grid generator. This generator  
 553 generates a torus with the given number of vertices, all with the same degree.
- 554 3. Scale-free graphs, which are used to model many social networks or web networks.  
 555 Graphstream Barabasi-Albert generator is used. This generator adds a vertex to the  
 556 graph and connects it to one or several vertices randomly chosen using Barabasi-Albert's  
 557 [1] preferential attachment rule. This operation is repeated for each vertex added.

558 Once the underlying graph is generated, the dynamic is introduced through a Markov  
 559 model (edge-Markovian dynamic graphs, introduced by [8]): edges appear and disappear  
 560 independently, and from the model two parameters are derived. The *presence* is equal to  
 561 the stationary probability of edge presence, often noted  $\pi_1$  ( $\pi_1 \in [0, 1]$ ). The *stability* of  
 562 the dynamic graph may be defined as the average number of steps during which one edge  
 563 remains present. A presence of 0.9 was chosen in our experiments (each edge is present 90%  
 564 of the time), for a stability equal to 9.

565 Once the dynamic graph is generated, the terminal vertices are selected randomly in the  
 566 vertex set.

567 Two experiments were run on randomly generated graphs. The first one aims at  
 568 analyzing the behavior of the algorithm (both through the number of solutions found  
 569 and the computation time) as it is computing time step by time step. The second experiment  
 570 aims at analyzing the outcome of the algorithm depending on the number of vertices of the  
 571 graph. Table 1 presents the parameters of the generated graphs for each experiment, where  
 572  $n$  is the number of vertices,  $k$  the size of the Steiner set,  $s$  the number of terminals and  $L$   
 573 the lifetime of the dynamic graph.

574 **5.1.2 Experiment on the behavior of the algorithm with regard to the  
575 number of iterations**

576 The curves are only presented for  $s = 3$ , but the results for other values are summarized in  
 577 table 2.

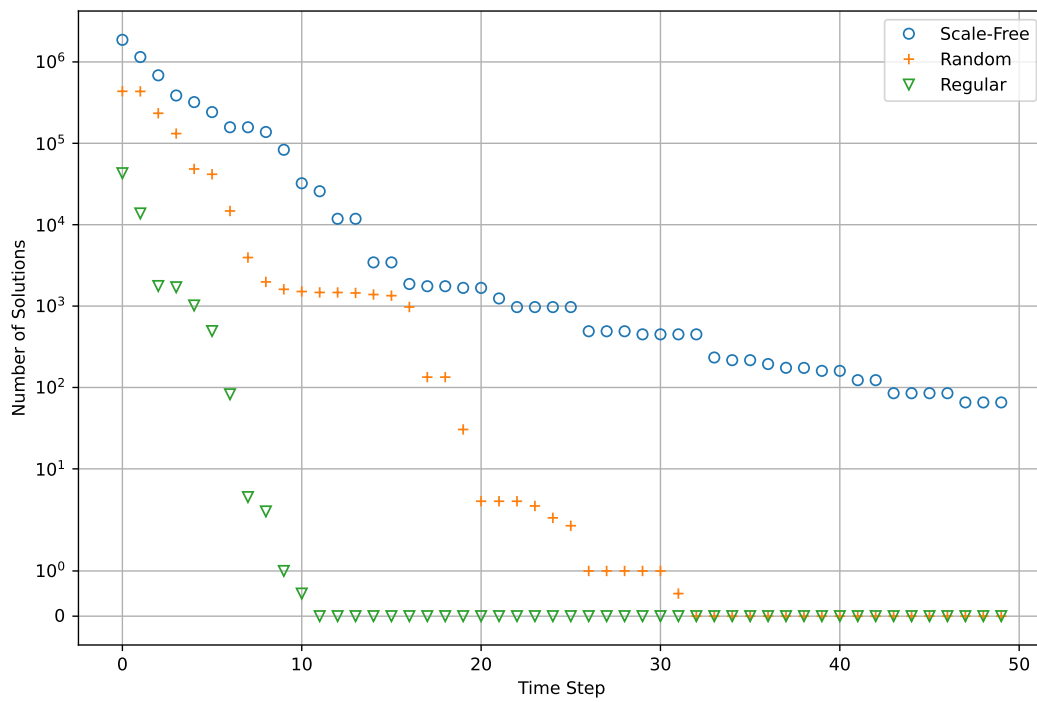
578 Figure 7 presents the evolution of the number of solutions when the number of iterations  
 579 (time steps) increases. The decrease is very fast for grids, and to a lesser extend for random

---

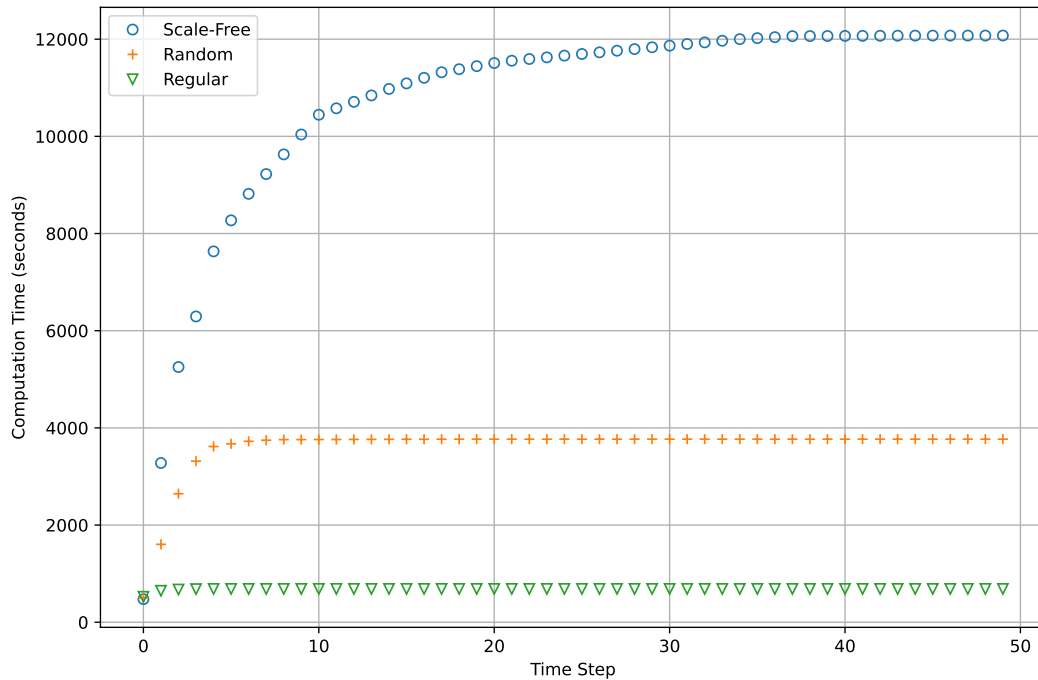
<sup>2</sup> <http://graphstream-project.org>

	First experiment	Second experiment
Graph type	Random, Scale-free, Regular	Random, Scale-free, Regular
n	49	36 ; 49 ; 64 ; 81 ; 100
k	10	10
S	3 ; 4 ; 5 ; 6	4
Presence	0.9	0.9
Stability	9	9
Average degree	4	4
T	50	50

■ **Table 1** Graphs generation settings



■ **Figure 7** Evolution of the median number of Steiner sets with regard to the number of time steps, with 3 terminal vertices.



■ **Figure 8** Evolution of the median computation time as it is being computed, with 3 terminal vertices.

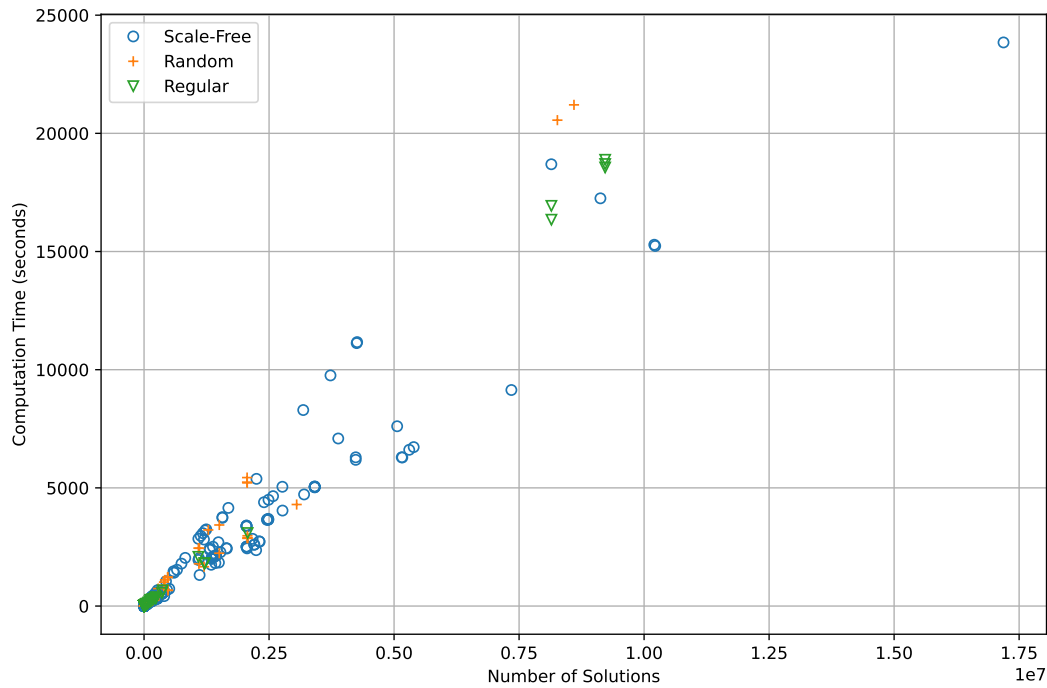
580 graphs. Indeed, it is very difficult to keep large persistent connected components (vertex sets  
 581 than remain connected) for these two classes of graphs, see[20]. So the number of Steiner  
 582 sets rapidly reaches 0. For scale free graphs however, the number of Steiner sets decreases  
 583 but much slower, and remains around 100 even for nearly 50 iterations. In these graphs,  
 584 there are some hubs with large degree that are very good candidates for the Steiner set.

585 Figure 8 provides the evolution of the total computation time. It is quite logically a  
 586 concave function that is constant as soon as the number of solution is 0. The theoretical  
 587 results on the algorithm complexity are confirmed by figure 9 that shows the computation  
 588 time per iteration varies linearly with the number of tested solutions.

### 589 5.1.3 Experiment on the runtime depending on the number of vertices 590 of the graph

591 Figure 10 shows that the complexity curve fits well with the regression function  $n^{k-s}$ , as  
 592 predicted by the algorithm analysis.

593 The main conclusion from this experimental study is that the algorithm can be applied  
 594 with medium size instances despite the high theoretical complexity. Another conclusion  
 595 concerns the number of possible Steiner sets. This number is highly dependent on the type  
 596 of underlying graph (footprint) and might decrease very fast with the number of iterations if  
 597 the graph is not scale free. These results were obtained for a moderate dynamic: a present  
 598 edge has a probability 0.9 to remain present from one iteration to the next.



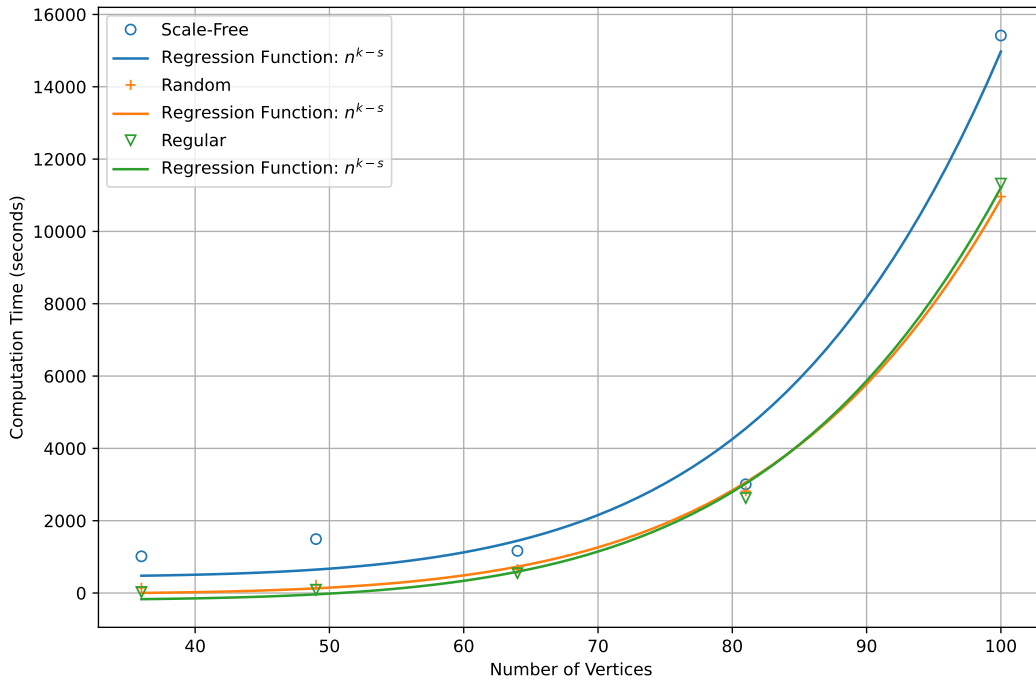
■ **Figure 9** Median computation time per iteration according to the number of tested vertex subsets, with 3 terminal vertices.

■ **Table 2** Number of solutions

Graph Type	S	Maximum Value (date $t = 0$ )	Minimum Value	First snapshot with Minimum Value
Scale-free	3	1 867 447,5	65,5	47
	4	256 030,0	5,0	47
	5	2 626,5	0,0	13
	6	2 405,5	0.0	17
Random	3	435 853,5	0.0	32
	4	33 258,5	0.0	17
	5	1 373,5	0.0	8
	6	85,0	0.0	3
Regular	3	42 909,5	0.0	11
	4	7 587,5	0.0	9
	5	10.0	0.0	4
	6	0.0	0.0	0

Scale-Free	0,976
Random	0,999
Regular	0,998

■ **Table 3**  $R^2$  values of the regression functions from Figure 10.



■ **Figure 10** Median computation time according to the number of vertices in the graph, compared with a regression function of the form  $n^{k-s}$ , which is  $n^6$  here.

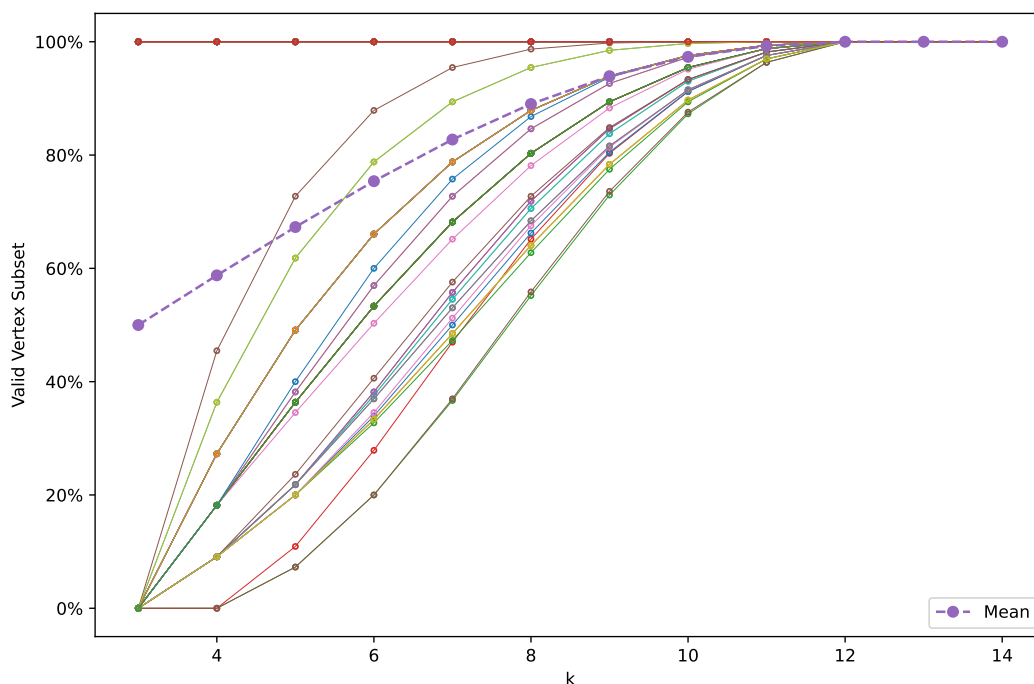
## 599 5.2 Real graphs

### 600 5.2.1 Experimental settings

601 The CRAWDAD VT/MANIAC dataset[13], accessible through the IEEE Dataport comes  
 602 from the CRAWDAD collection. This dataset encompasses routing and topology traces  
 603 gathered during the Mobile Ad hoc Networks Interoperability And Cooperation (MANIAC)  
 604 Challenges that took place in 2007 and 2009 in conjunction with the IEEE Globecom IEEE  
 605 and PerCom conferences. These traces provide insights into the communication patterns,  
 606 node mobility, and network structure characteristic of MANETs (Mobile Adhoc Networks)  
 607 in real-world scenarios.

608 The dataset comprises log files providing information for each monitored device. These  
 609 files can be interpreted as routing tables for each device on a per-second basis. Based on this  
 610 information, a dynamic graph has been constructed, where nodes represent mobile stations  
 611 and edges depict connections between two mobile stations. The edges are labeled with a list  
 612 of timestamps indicating the dates when these connections are effective. In the presented  
 613 results, the experiment involves 14 mobile stations for a duration of about 20 minutes. It  
 614 translates into a graph with 14 nodes, 74 edges (for the footprint), and 1244 timesteps. On  
 615 average, edges are present for 592 seconds (less than 10 minutes) during the experiment,  
 616 which is 48% of the time. When present, edges remain up for 114 seconds (less than 2  
 617 minutes) on average (9.1% of the time). In comparison to the artificial graphs used in the  
 618 previous experiment, the *presence* is lower here (48%) than before (90%).

619 If for some snapshot, the terminals can not be connected, the number of Steiner sets is  
 620 of course 0. Hence it is useful to verify this first. An efficient algorithm to compute open  
 621 Eternal Connected Components *oECC* is available, see[20]. After running this algorithm,



■ **Figure 11** Real dataset: percentage of Steiner sets found for different values of  $k$  ( $s = 4$ ).

622 an  $oECC$  with 9 vertices has been found. If all terminal vertices are inside this  $ECC$ , it  
 623 guarantees the existence of at least one Steiner set (of size  $n - s$ ). Conversely, no Steiner set  
 624 exists if the terminal set is not included into an  $oECC$ . Note however that a vertex outside  
 625 the  $oECC$  may still belong to some Steiner set, due to the open nature of the component.  
 626 We decided to look only for vertices of the  $oECC$  to build the terminal set, whereas the  
 627 Steiner set candidates are included into  $V$ .

## 628 5.2.2 Results and discussion

629 Figure 11 shows the percentage of Steiner sets among all possible vertex sets of same size,  
 630 for all possible terminal sets of size  $s = 3$  (one line may represent different terminal sets,  
 631 with the same behavior). The same kind of "banana" curve has been obtained for  $s = 4, 5, 6$ .  
 632 Note the large value of the mean, hence the large number of cases where the terminal set is  
 633 already connected ( $x$ -axis = 3,  $y$ -axis = 100%), due to the nature of the graph: a bit less  
 634 than 50%. Conversely, there is almost always at least one solution (except in some cases  
 635 when  $k = s + 1 = 4$ ). The results are similar for larger values of  $s$ .

636 So it is nearly always possible to find a Steiner set in this example, for any value of  $s$  and  
 637  $k$  except specific cases with  $k - s = 1$  or  $k - s = 0$ . The finding is very fast (a few seconds at  
 638 most) even if the time horizon is very large.

## 639 6 Conclusion

640 In this paper, we presented several possible extensions of the Steiner Tree problem to the  
 641 case of dynamic (or temporal) graphs. The goal is to maintain instantaneous connectivity  
 642 between all terminal vertices for all snapshots. The chosen extension minimizes the size of



643 the Steiner set. It can be further extended to minimize the sum of the costs of the edges used  
 644 at each time step as a the second criterion. We proved that the associated decision problem  
 645 is NP-Complete even when the costs are not considered, and there are only two terminals.  
 646 The proof is rather straightforward thanks to a reduction to Vertex Cover. [However, we also](#)  
 647 [prove that this remains true even when the lifetime is 2, through another and a bit more](#)  
 648 [complicated reduction from SAT.](#)

649 This paper proposed an exact algorithm that computes all Steiner sets of a given size,  
 650 studied its complexity, and provided some experimental tests. The algorithm may be used  
 651 on-line. First, randomly generated dynamic graphs of different types have been used. The  
 652 algorithm was able to solve small to medium size instances. The number of solutions, hence  
 653 the computation time, is highly dependent on the type of graphs. The results confirmed  
 654 Steiner sets are much more numerous on scale free graphs. Second, the algorithm has been  
 655 applied to a dynamic graph obtained from real data considering MANET's. The underlying  
 656 graph is small, but the time horizon is large. The algorithm was able to find rapidly all  
 657 Steiner sets, for different parameter values. [Of course the efficiency will be very versatile](#)  
 658 [according to the type of applications, that use graphs of very different kinds and very different](#)  
 659 [sizes.](#)

---

## 660      **References**

- 661    1    Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews*  
 662    *of modern physics*, 74(1):47–97, 2002.
- 663    2    Kyriakos Axiotis and Dimitris Fotakis. On the size and the approximability of minimum  
 664    temporally connected subgraphs. In *43rd International Colloquium on Automata, Languages,*  
 665    *and Programming (ICALP 2016)*, 2016. arXiv preprint arXiv:1602.06411.
- 666    3    Stefan Balev, Eric Sanlaville, and Jason Schoeters. Temporally connected components.  
 667    *Theoretical Computer Science*, 1013:114757, 2024.
- 668    4    Sandeep Bhadra and Afonso Ferreira. Computing multicast trees in dynamic networks and  
 669    the complexity of connected components in evolving graphs. *Journal of Internet Services and*  
 670    *Applications*, 3:269–275, 2012.
- 671    5    Robert Bredereck, Christian Komusiewicz, Stefan Kratsch, Hendrik Molter, Rolf Niedermeier,  
 672    and Manuel Sorge. Assessing the computational complexity of multilayer subgraph detection.  
 673    *Network Science*, 7(2):215–241, 2019.
- 674    6    Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying  
 675    graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed*  
 676    *Systems*, 27(5):387–408, 2012.
- 677    7    Arnaud Casteigts, Joseph G Peters, and Jason Schoeters. Temporal cliques admit sparse  
 678    spanners. *Journal of Computer and System Sciences*, 121:1–17, 2021.
- 679    8    Andrea EF Clementi, Claudio Macci, Angelo Monti, Francesco Pasquale, and Riccardo Silvestri.  
 680    Flooding time in edge-markovian dynamic graphs. In *Proceedings of the twenty-seventh ACM*  
 681    *symposium on Principles of distributed computing*, pages 213–222, 2008.
- 682    9    Antoine Dutot, Frédéric Guinand, Damien Olivier, and Yoann Pigné. Graphstream: A tool  
 683    for bridging the gap between complex systems and dynamic graphs. In *Emergent Properties*  
 684    *in Natural and Artificial Complex Systems. Satellite Conference within the 4th European*  
 685    *Conference on Complex Systems (ECCS'2007)*, 2007.
- 686    10    Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung.*  
 687    *Acad. Sci*, 5(1):17–60, 1960.
- 688    11    Lester R. Ford and Delbert R. Fulkerson. Constructing maximal dynamic flows from static  
 689    flows. *Operations Research*, 6(3):419–433, 1958.
- 690    12    Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. Freeman  
 691    San Francisco, 1979.

- 692 13 Amr Hilal, Jawwad N Chattha, Vivek Srivastava, Michael S Thompson, Allen B MacKenzie,  
693 Luiz A DaSilva, and Pallavi Saraswati. *Crawdad vt/maniac*, 2022. URL: <https://dx.doi.org/10.15783/C7WG6T>, doi:10.15783/C7WG6T.
- 694  
695 14 Mark Jones, Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Ondřej Suchý.  
696 Parameterized complexity of directed steiner tree on sparse graphs, 2012. [arXiv:1210.0260](https://arxiv.org/abs/1210.0260).
- 697 15 Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US,  
698 Boston, MA, 1972. doi:10.1007/978-1-4684-2001-2\_9.
- 699 16 Nina Klobas, George B Mertzios, Hendrik Molter, and Paul G Spirakis. The complexity of  
700 computing optimum labelings for temporal connectivity. *Journal of Computer and System*  
701 *Sciences*, 146:103564, 2024.
- 702 17 Othon Michail and Paul G Spirakis. Traveling salesman problems in temporal graphs.  
703 *Theoretical Computer Science*, 634:1–23, 2016.
- 704 18 Martin Skutella. An introduction to network flows over time. In *Research Trends in*  
705 *Combinatorial Optimization*, pages 451–482. Springer, 2009.
- 706 19 Mathilde Vernet, Maciej Drozdowski, Yoann Pigné, and Eric Sanlaville. A theoretical and  
707 experimental study of a new algorithm for minimum cost flow in dynamic graphs. *Discrete*  
708 *Applied Mathematics*, 296:203–216, 2021.
- 709 20 Mathilde Vernet, Yoann Pigne, and Eric Sanlaville. A study of connectivity on dynamic  
710 graphs: computing persistent connected components. *4OR*, 21(2):205–233, 2023.
- 711 21 Kevin White, Martin Farber, and William Pulleyblank. Steiner trees, connected domination  
712 and strongly chordal graphs. *Networks*, 15(1):109–124, 1985.
- 713 22 B Bui Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost  
714 journeys in dynamic networks. *International Journal of Foundations of Computer Science*,  
715 14(02):267–285, 2003.