

On the Feasibility of Maintenance Algorithms in Dynamic Graphs

Arnaud Casteigts¹, Bernard Mans², and Luke Mathieson²

¹ SITE, University of Ottawa, 800 King Edward, Ottawa, Ontario K1N 6N5, Canada. casteig@site.uottawa.ca

² Department of Computing, Macquarie University, Sydney, NSW 2109, Australia. {bernard.mans,luke.mathieson}@mq.edu.au

Abstract. Near ubiquitous mobile computing has led to intense interest in dynamic graph theory. This provides a new and challenging setting for algorithmics and complexity theory. For any graph-based problem, the rapid evolution of a (possibly disconnected) graph over time naturally leads to the important complexity question: is it better to calculate a new solution from scratch or to adapt the known solution on the prior graph to quickly provide a solution of guaranteed quality for the changed graph?

In this paper, we demonstrate that the former is the best approach in some cases, but that there are cases where the latter is feasible. We prove that $W[1]$ -hardness for the parameterized approximation problem implies the non-existence of a maintenance algorithm for the given approximation ratio, even given time exponential in the solution size and polynomial in the over all size — i.e., even with a large amount of time, having a solution to a very similar graph does not help in computing a solution to the current graph. To achieve this, we formalize the idea as a *maintenance algorithm*. To illustrate our results we show that r -REGULAR SUBGRAPH is $W[1]$ -hard for the parameterized approximation problem and thus has no maintenance algorithm for the given approximation ratio. Conversely we show that VERTEX COVER, which is fixed-parameter tractable, has a 2-approximate maintenance algorithm. The implications of NP -hardness and NPO -hardness are also explored.

1 Introduction

With the development of sufficiently small and powerful hardware, mobile computing devices present a new and interesting network environment in which the complexity and algorithmics for even well understood problems can change radically. To model this sort of network, various notions of dynamic graphs have been developed, including *delay-tolerant* [7], *disruptive-tolerant*, *intermittently-connected* [20], *opportunistic*, *time-varying* [4] and *evolving* [2,3]. Each of these models differing aspects of the dynamics under differing assumptions or with different applications in mind. One of the key aspects of dynamic graphs is that a graph may not be connected at any given moment, or ever, but due to the appearance and disappearance of edges, it may still be possible to construct

journeys (rather than paths) through the graph over time and space. However, dependent on the assumptions of the system, there may be even fundamental properties or traditional problems which are incomputable or definitionally ambiguous (e.g., [2]). The results presented here are independent of the model as long as the change in the graph occurs (or can be described) in a discrete manner.

Given a traditional graph problem Π and a sequence of graphs $\{G_i\}$, we may reframe Π in several ways as a dynamic graph problem; *permanent* — where we try to find a solution that holds at all points in time (e.g., a single set of dominators that cover all nodes in every G_i), *over-time* — where the solution is defined with respect to the sequence as a whole (e.g., a set of dominators such that each node is covered in at least one G_i), and *evolving* — where we compute a (possibly different) solution for each G_i . Permanent and over-time problems may appear somewhat less general in that they require prior knowledge of the graph dynamics. However it is worth noting that there are relevant in few, yet important, practical scenarios, such as with a known schedule (e.g., public transports, low-earth satellites, sensors with sleeping schedule), or a known schedule property (e.g., *periodicity* [8] or *recurrence* [4] — i.e., edges that exist once are guaranteed to re-appear at some known, bounded, or unbounded time).

It can be observed that for some problems, e.g., *covering* problems like VERTEX COVER, all three variants are strongly related. Given such a problem Π , one can easily check that a solution to *permanent* Π is a (possibly far from optimal) solution to *evolving* Π , and a solution to any G_i in the *evolving* Π is also valid for *over-time* Π . The connexion is even stronger: the *intersection* of solutions to *evolving* Π is valid for *over-time* Π , and their *union* is a solution to *permanent* Π . From this perspective, the *evolving* variant appears quite central, and the *permanent* or *over-time* Π actually form upper and lower bounds for each solution in the *evolving* Π . In this paper, we focus on the *evolving* Π and look at finding the solution for each point in time.

The dynamic context is an interesting algorithmic and complexity setting. Although a naïve approach would simply compute a new solution for every G_i , it is possible that given a relatively limited amount of change in the graph at each step we may leverage the previous solution to allow quick computation of the new solution. This leads to the idea of a *maintenance algorithm* (q.v., Section 2). Moreover this approach fits well with the real world inspiration, where along with a possibly rapid pace of change, there is often no centralised control. If the previous solution can be used to compute the new solution, this suggests a certain localization, which would be ideal for mobile infrastructureless networks. Considering that dynamic graphs can be permanently disconnected (while still offering connectedness over time), we initially consider well-known problems that are not ambiguous or undefined in a disconnected setting: r -REGULAR SUBGRAPH and VERTEX COVER. A solution must be well defined for each (possibly) partitioned component and trivial solutions must exist for trivial cases (e.g., single nodes).

As we are mainly interested in maintaining a solution of some guaranteed quality for each particular problem, our results can be summarized as follows:

- we prove that $W[1]$ -hardness for the parameterized approximation problem implies the non-existence of a maintenance algorithm for the given approximation ratio, even given time exponential in the solution size and polynomial in the over all size,
- we show that r -REGULAR SUBGRAPH and several variants have no parameterized approximation algorithms thus have no FPT -time maintenance algorithm, and conversely,
- we show that problems that are fixed-parameter tractable may have an approximate maintenance algorithm by proving that this is indeed the case for VERTEX COVER,
- we establish a complexity classification for maintenance algorithms that provides a strong relationship with parameterized approximation complexity [12].

Similar ideas regarding dynamic algorithms and complexity have been explored before, but from notably different positions. Typically previous work has focussed on polynomial or logarithmic time problems, and did not include the consideration of locality that is central to many practical applications for dynamic complexity (central oversight and communication is neither guaranteed nor generally desirable in a dynamic network), Section 2 gives the relevant definitions. However many interesting results and ideas are applicable in this context. Patnaik & Immerman [15] consider dynamic complexity from a descriptive complexity theory perspective, defining DYNFO, a class of dynamic problems that are expressible in first order logic. Weber & Schwentick [19] build upon this, again concentrating on a descriptive complexity approach. Holm, de Lichtenberg & Thorup [10] give a series of results that can readily be interpreted as maintenance algorithms in our context. Their results for CONNECTIVITY, 2-EDGE and BICONNECTIVITY rely on the maintenance under edge deletion and addition of a solution for MINIMUM SPANNING FOREST, giving polylogarithmic running times for all problems, but with no bound on locality. Miltersen *et al.* [14] present another, similar approach where the dynamism is achieved at a lower level by perturbing individual bits in the input. They also focus on problems of polynomial complexity showing, in our context, that problems such as the CIRCUIT VALUE PROBLEM and PROPOSITIONAL HORN SATISFIABILITY have no polylogarithmic maintenance algorithms but that interestingly there exist other P -complete problems that do. Ausiello, Bonifaci & Escoffier [1] discuss a different model, where there is only interest in using an existing optimal solution to solve (to some degree of approximation) a perturbed instance (the model is generally called *reoptimization*). This approach is in some ways an unsuitable perspective for our context, as we do not expect at any point to be able to compute an optimal solution, however their negative results in particular carry to our setting, such as MIN COLORING having no P -time maintenance algorithm for approximation ratio $\frac{4}{3} - \varepsilon$ for any ε .

The rest of the paper is organized as follows. We define our notation and provide our main results in Section 2. We provide some basic and necessary background on parameterized complexity and parameterized approximation in

Section A. We provide the main proofs of our results and of results on parameterized approximation complexity for VERTEX COVER, r -REGULAR SUBGRAPH and some generalizations of r -REGULAR SUBGRAPH in Section 3. We give some concluding remarks in Section 4.

2 The Feasibility of Maintenance

The key difference in the dynamic setting with regards to computation is that as the graph is continually changing, we are required to continually recompute solutions for the problem. Hence the style of algorithm we are concerned with is somewhat different. In this case we would ideally like an algorithm that can exploit the known solution to the previous graph, which is somewhat similar to the current graph, to produce a solution for the current graph that maintains a desired solution quality (whether optimal or within some approximation bound) in a time faster than it would take to completely recompute the solution. We formalize the idea as a *maintenance algorithm*.

Definition 1 (Maintenance Algorithm). *Given a sequence of graphs $\{G_i\}$ where the editing distance between G_i and G_{i+1} is 1, \mathcal{A} is a maintenance algorithm for problem Π if given a solution S_i for Π on graph G_i , \mathcal{A} can compute a solution S_{i+1} for Π on graph G_{i+1} that preserves the quality of the solution.*

The *editing distance* between two graphs is the number of *editing operations* that have to be performed to obtain one from the other. Typically the editing operations we will be interested in are some subset of edge addition, edge deletion, vertex addition and vertex deletion. We set the editing distance here to 1 in order to remain sufficiently general in establishing *negative* results.

As the graph changes relatively quickly, we would like the computation to be done within a limited computation time (preferably a small constant). The notion of a *bounded* maintenance algorithm formalizes this bound.

Definition 2 (Bounded Maintenance Algorithm). *A maintenance algorithm is bounded if there is a function f such that at each step, the computation performed at each vertex is time bounded by f . We refer to such an algorithm as a f -maintenance algorithm.*

Also, because in a practical setting the nodes model independent devices, we would like the computation to be done as locally as possible, *i.e.*, the state of each vertex being based only on the state of its r -neighborhood, with $r \ll \text{diameter}(G)$.

Definition 3 (r -Local Maintenance Algorithm). *A maintenance algorithm is r -local if the state of each vertex is computed based only on the states of vertices within distance r . We refer to such an algorithm as a r -local maintenance algorithm.*

For example a 1-local $O(1)$ -maintenance algorithm allows each vertex to perform a constant number of computational steps at each iteration of the algorithm, knowing only the states of its direct neighbors. It is with these computational restrictions that knowledge of a prior solution and the concept of a maintenance algorithm can be most useful. Indeed there are many prior results that fit into this framework, though previously there has been little consideration of the locality. For example,

Lemma 1 ([14]). *UNDIRECTED FOREST ACCESSIBILITY has a d -local $O(\log n)$ -maintenance algorithm where d is the diameter of the graph and n is the number of vertices.*

Lemma 2 ([10]). *MINIMUM SPANNING FOREST has a d -local $O(\log^4 n)$ -maintenance algorithm where d is the diameter of the graph.*

Lemma 3 ([17]). *DIRECTED REACHIBILITY has a d -local $O(m + n \log n)$ -maintenance algorithm where d is the diameter of the graph, m is the number of edges and n is the number of vertices.*

Ausiello, Bonifaci & Escoffier [1] examine the sort of harder problems that we are interested in (rather than P -time problems), however their setting requires an initially optimal solution and only covers a single perturbation, rather than the highly dynamic environment expected from the application domain we consider.

Of course there are limits to the existence of maintenance algorithms:

Theorem 1. *For any computable function f and polynomial p , if a problem Π is $W[t]$ -hard for parameter k and any $t \geq 1$, Π has no $O(f(k)p(n))$ -maintenance algorithm unless $W[t] = FPT$ where n is the size of the input.*

Proof. Assume that Π has such an algorithm.

Then let (G, k) be an instance of Π and G_0 be the trivial instance defined by taking the completely disconnected graph on $|V(G)|$ vertices with trivial optimal solution S_0 . Let $G_0, \dots, G_{|E(G)|}$ be a sequence of graphs generated by adding the edges of G one by one in arbitrary order to G_0 , resulting in $G_{|E(G)|} = G$.

As Π has a maintenance algorithm beginning with G_0 the algorithm can be applied to each pair G_i, G_j where $j = i + 1$ with solution S_i to obtain solution S_j . Then if $S_{|E(G)|}$ is a witness that (G, k) is a YES-instance, the algorithm answers YES (or returns $S_{|E(G)|}$ in the case of a search problem) and NO otherwise.

Then we have an algorithm that performs $O(f(k)p(n))$ operations for each vertex at each iteration. As there are at most n^2 iterations, the overall algorithm has running time $O(n^3 f(k)p(n))$, and hence Π is fixed-parameter tractable, which contradicts the $W[t]$ -hardness of Π with parameter k .

By the same argument, we can also obtain a classical complexity analogue.

Corollary 1. *For any polynomial p , if a problem Π is NP-hard, Π has no $O(p(n))$ -maintenance algorithm that optimally solves Π unless $P = NP$ where n is the size of the input.*

Similarly, classical approximation results are preserved.

Corollary 2. *For any polynomial p , if a problem Π is NPO - PB -hard, Π has no $O(p(n))$ -maintenance algorithm that solves Π within an approximation factor of $O(n^{1-\varepsilon})$ for any $\varepsilon > 0$ unless $P = NP$ where n is the size of the input.*

If, given a parameterized problem with parameter k , the bound is $O(f(k)p(n))$ where f is a computable function, p is a polynomial in n , the size of the input, we denote the associated bounded maintenance algorithm as an (r -local) FPT -maintenance algorithm. The complexity results given in Section 3 then give the following results.

Theorem 2. $g(k)$ -APPROX-VERTEX DELETION TO REGULAR SUBGRAPH has no FPT -maintenance algorithm unless $W[1] = FPT$.

Theorem 3. $g(k)$ -APPROX-DELETION TO REGULAR SUBGRAPH has no FPT -maintenance algorithm unless $W[1] = FPT$.

Theorem 4. $g(k)$ -APPROX-WEIGHTED DEGREE CONSTRAINED DELETION has no FPT -maintenance algorithm unless $W[1] = FPT$.

Similar statements can be made regarding c -ADD-APPROX-DOMINATING SET and $g(k)$ -APPROX-INDEPENDENT DOMINATING SET using their respective approximation hardness results:

Lemma 4 ([6]).

c -ADD-APPROX-DOMINATING SET has no FPT -maintenance algorithm unless $W[2] = FPT$.

Lemma 5 ([6]). $g(k)$ -APPROX-INDEPENDENT DOMINATING SET has no FPT -maintenance algorithm unless $W[1] = FPT$.

Independently of the results above, we observe that if the bound on the maintenance algorithm is further restricted, then in certain cases no approximation ratio can be maintained. More precisely the approximation ratio will be a function of the number of iterations of the maintenance algorithm, up to trivial bounds on the quality of the solution. Given a (w.l.o.g) minimization problem Π in NPO with the following properties:

1. The decision counterpart of Π is NP -hard.
2. Π has a $O(1)$ -maintenance algorithm \mathcal{A} .

We make the following claims:

Lemma 6. *There is a step in \mathcal{A} that is divergent.*

Where *divergent* means that if the size of the optimal solution decreases, the size of the solution given by \mathcal{A} does not decrease, and similarly if the size of the optimal solution does not change, the size of the solution given by \mathcal{A} increases.

Proof. Follows immediately from the NP -hardness of the decision counterpart of Π .

Let γ_i be the size of the solution given as input from instance I_i (perhaps as the result of a previous application of \mathcal{A}) and γ_i^* be the size of the optimal solution for I_i . Let I_{i+1} be the instance after alterations and γ_{i+1} be the size of the solution given by \mathcal{A} and γ_{i+1}^* be the size of the optimal solution.

Lemma 7. *Given an approximation ratio A and an instance with solution of size $\gamma_i = A \cdot \gamma_i^*$, then \mathcal{A} cannot guarantee that $\gamma_{i+1} \leq A \cdot \gamma_{i+1}^*$.*

Proof. Let the change from I_i to I_{i+1} be such that it induces \mathcal{A} to take a divergent step. Assume without loss of generality that $\gamma_{i+1} = \gamma_i + 1$ and $\gamma_{i+1}^* = \gamma_i^*$. Then

$$\frac{\gamma_{i+1}}{\gamma_{i+1}^*} = \frac{\gamma_i + 1}{\gamma_i^*} = \frac{A \cdot \gamma_i^* + 1}{\gamma_i^*} = A + \frac{1}{\gamma_i^*}$$

It is easy to see that a similar result occurs with the other possibilities for divergence.

Given a problem Π with a *long divergent sequence* this problem may be amplified. A *long divergent sequence* is a set of instances where the changes between each instance induce divergent steps in \mathcal{A} . For example given the completely disconnected graph and the problem DOMINATING SET, the initial solution is to take all vertices in the dominating set (which is optimal), then one by one we add edges to obtain a star graph. Using an adversarial approach we add the edges such that the centre of the star is not in the dominating set, but all other vertices are, taking $n - 1$ vertices where only 1 is needed.

Lemma 8. *Given a long divergent sequence of length d , we have $\gamma_{d+1}/\gamma_{d+1}^* \geq 1 + d/\gamma_{d+1}^*$.*

Proof. At each step we are forced to increase the size of the solution relative to the optimal by at least one. Then by step d (instance I_{d+1}) we have $\gamma_{d+1} \geq \gamma_1 + d$ however $\gamma_{d+1}^* \leq \gamma_1^*$.

Therefore we at least have:

$$\frac{\gamma_{d+1}}{\gamma_{d+1}^*} \geq \frac{\gamma_1 + d}{\gamma_1^*} = \frac{\gamma_1^* + d}{\gamma_1^*} = 1 + \frac{d}{\gamma_1^*} \geq 1 + \frac{d}{\gamma_{d+1}^*}$$

Then for the dominating set example we have the approximation ratio of $O(n)$. In general, given a problem with a long divergent sequence of length $O(n)$, we obtain a similar approximation ratio.

3 Problems in Dynamic Graphs

Given a graph, the problem of obtaining an r -regular subgraph with a minimum number of excluded vertices (known variously as r -REGULAR SUBGRAPH,

k -ALMOST r -REGULAR SUBGRAPH and VERTEX DELETION TO REGULAR SUBGRAPH) is $W[1]$ -hard [13]. The property of having a bounded degree or regular graph is particularly interesting for routing purposes. Although it is $W[1]$ -hard, for a dynamic graph, an approximation would be sufficient as we expect the graph to change rapidly, so computing a reasonable solution quickly is more effective than computing an exact solution slowly. In this section however we show that this problem has no parameterized approximation unless $W[1] = FPT$.

Formally we define the problem as:

$g(k)$ -APPROX-VERTEX DELETION TO REGULAR SUBGRAPH

Instance: A graph $G = (V, E)$, integers k and r .

Parameter: k .

Question: Is there a set $V' \subset V$ with $|V'| \leq k$ such that the subgraph $G' = G[V \setminus V']$ is r regular?

This problem has many possible generalizations, we note two in particular:

$g(k)$ -APPROX-DELETION TO REGULAR SUBGRAPH

Instance: A graph $G = (V, E)$, integers k and r .

Parameter: k .

Question: Is there a set $D \subset V \cup E$ with $|D| \leq k$ such that the subgraph $G' = (V \setminus D, E \setminus D)$ is r regular?

$g(k)$ -APPROX-WEIGHTED DEGREE CONSTRAINED DELETION

Instance: A graph $G = (V, E)$, integers k and r , a weight function $\rho : V \cup E \rightarrow [1, k + 1]$ and a degree list function $\delta : V \rightarrow 2^{\{0, \dots, r\}}$.

Parameter: k .

Question: Is there a set $D \subset V \cup E$ with $\sum_{d \in D} \rho(d) \leq k$ such that for every vertex v in the subgraph $G' = (V \setminus D, E \setminus D)$ we have $\sum_{e \in N^{G'}(v)} \rho(e) \in \delta(v)$?

The reduction we use is from the following problem:

STRONGLY REGULAR MULTICOLORED CLIQUE

Instance: A graph $G = (V, E)$ with $V = \uplus_{i \in [k]} V_i$ such that $|V_i| = s$ and for all $v \in V_i$ we have $d(v)|_{V_j} = d$ for all i, j , and an integer k .

Parameter: k .

Question: Is there a clique V' of size k such that V' has one vertex from each V_i ?

In the remainder of this section we present the hardness reduction that demonstrates that $g(k)$ -APPROX-VERTEX DELETION TO REGULAR SUBGRAPH and subsequently the more general problems are $W[1]$ -hard, and therefore have no FPT -maintenance algorithms. We then prove that problems that are fixed-parameter tractable may have an approximate maintenance algorithm by proving that this is indeed the case for VERTEX COVER.

3.1 Hardness Results

Lemma 9. $g(k)$ -APPROX-VERTEX DELETION TO REGULAR SUBGRAPH is $W[1]$ -hard.

Proof. The underlying exact problem VERTEX DELETION TO REGULAR SUBGRAPH was shown to be $W[1]$ -hard in [13] with a reduction that can be used with little modification to show hardness for the approximation problem.

The reduction is from STRONGLY REGULAR MULTICOLORED CLIQUE. Let (G, k) of STRONGLY REGULAR MULTICOLORED CLIQUE where $V(G) = \uplus_{i \in [k]} V_i$ is partitioned into k color classes where $|V_i| = s$ for all i and each vertex has d neighbours in each color class. We construct an instance (G', k') of $g(k)$ -APPROX-VERTEX DELETION TO REGULAR SUBGRAPH by setting $k' = k + \binom{k}{2}$, creating a vertex copy V'_i for each color class V_i and a set P_{ij} of vertices for each pair V_i, V_j of color classes, which will control edge selection. We make each V'_i a complete graph. For every pair of vertices u, v with $u \in V_i$ and $v \in V_j$, $i \neq j$, let u' and v' be the corresponding vertices in V'_i and V'_j . If $uv \in E(G)$ we add two vertices $u'_{v'}, v'_{u'}$ to P_{ij} with the edges $u'u'_{v'}$, $u'_{v'}v'_u$ and $v'v'_{u'}$. For each pair of vertices u, u' in P_{ij} where u and u' are in the same V_l ($l = i, j$) and $u \neq u'$ we add the edge uu' . We choose r to be greater than $\max\{(s-1) + d(k-1), 2 + (s-1)d\}$ such that r is also of opposite parity to s , the smallest such r suffices. Then for each V'_i we add a set of vertices V''_i where each vertex in V''_i has an edge to each vertex in V'_i such that the degree of each vertex in V'_i is $r + 1$. We do similarly for each P_{ij} with a set P'_{ij} . We then increase the degree of each vertex in each V''_i and each P'_{ij} to $r + 1$ by taking an set of $r + 1$ clique, breaking an edge in each and attaching it at these two points to the vertex being adjusted. For each of these vertices, if the total number of vertices in the attached cliques is less than $g(k')$ we augment them by adding new $r + 1$ cliques by breaking an edge in the old and new clique and reattaching them to each other.

Claim. If G has a properly colored clique of size at least k , then we can delete at most k' vertices to make G' r -regular.

This claim follows as in [13]. Note in particular that G has a clique of size at most k (as there are only k color classes and we must use at least k' vertices, one from each V'_i and two from each P_{ij} to make the graph r -regular. The vertices chosen for deletion correspond to the vertices and edges of the clique.

Claim. If G' can be made r -regular by the deletion of at most $g(k')$ vertices, then G has a properly colored clique of size at least k .

Note that we must delete at least one vertex from each V'_i and two from each P_{ij} . If we delete any vertices from the degree adjustment components of the graph, V''_i , P'_{ij} or the adjustment cliques, then we must delete all such vertices, which would require more than $g(k)$ deletions. Therefore the k' vertices must be the only vertices deleted. As with [13], these must correspond to a properly colored k -clique in G .

Note also that if we allow edge deletion as well as vertex deletion, we obtain no advantage by deleting any edges, therefore any deletions must be of vertices only. This gives the following corollary.

Corollary 3. $g(k)$ -APPROX-DELETION TO REGULAR SUBGRAPH is $W[1]$ -hard.

By subproblem containment, the more general version where we allow each vertex to have a list of possible final degrees, rather than simple degree r is also $W[1]$ -hard.

Corollary 4. $g(k)$ -APPROX-WEIGHTED DEGREE CONSTRAINED DELETION is $W[1]$ -hard when the possible editing operations are vertex deletion or vertex and edge deletion.

3.2 A Maintenance Algorithm for Vertex Cover

Thus we may conclude that maintenance is not feasible (in the complexity guarantee sense) even if we have polynomial time at each step (in fact we may relax this perhaps to fixed-parameter tractable time, dependent on the number of steps), if the parameterized approximation problem is $W[t]$ -hard for any $t \geq 1$.

However if the parameterized approximation problem is fixed-parameter tractable, then we may have some hope. In a broad sense many kernelisation algorithms may be reinterpreted as maintenance algorithms, e.g.,

Lemma 10 ([16]). k -MAX CUT has an FPT-time $2k$ -local maintenance algorithm.

Lemma 11 (Attribute to Buss in [5]). VERTEX COVER has an FPT-time $2k^2$ -local maintenance algorithm.

Note that these examples were not chosen as necessarily the most efficient algorithms for these problem, but because they employ simple, local reduction rules. However in both cases (and all such adaptations) the algorithm does not use any prior solution, so perhaps is not the most effective maintenance strategy. Nonetheless they do give goals for the performance of any maintenance algorithm.

By allowing an approximate solution, we can develop potentially much simpler, faster and more local maintenance algorithms. We demonstrate with the following case, VERTEX COVER, which is fixed-parameter tractable, therefore fixed-parameter approximable and has a simple greedy 2-approximation. Furthermore there is evidence that a 2-approximation is in some sense the best possible ratio [11]. In this case, the following maintenance algorithm is efficient and of tolerable approximation quality.

Lemma 12. There is a 1-local maintenance algorithm for MAXIMAL MATCHING and hence there is a 1-local maintenance algorithm for VERTEX COVER with an approximation factor of 2.

Proof. The algorithm exploits the result that any maximal matching is a 2-approximation for the size of the minimum vertex cover.

Given a graph G , with vertex cover S , and a set of edge deletions and additions T , the algorithm proceeds by maintaining a matching at the site of the deletions and additions. We may assume that at each completed step in the algorithm (as this algorithm can be used to generate the initial solution) each non-isolated vertex in the vertex cover has a *pair-vertex*, its partner in the matching.

Then for each element t of T , if t is a deletion that separates a vertex from its pair-vertex, and the vertex has any neighbour not in the cover, we choose arbitrarily one of these neighbours to be its new pair-vertex. If it has no such neighbours, the vertex is removed from the cover.

Otherwise if t is an addition that connects two vertices not in the cover, they are both added to the cover and are noted as each others pair-vertex.

Then at any point in the algorithm each vertex in the cover has an adjacent pair-vertex (i.e., the edge between the two is a matching edge), and there is no edge that has neither endpoint in the cover. Therefore the implicit matching is maximal. Thus we immediately have our 2-approximation.

By way of contrast, MAX CUT admits a simple $\frac{1}{2}$ approximation algorithm [18] (though this is far from the best ratio) which is easily translated into a simple maintenance algorithm however the locality seems to be the diameter of the graph in the worst case. This suggests that there is potentially some separation possible between “local” and “non-local” problems that is not captured by regular complexity classes. This idea has been explored from a different perspective by Militeren *et al.* [14] who show a split of P -complete problems by the class *incr-POLYLOGTIME*.

4 Conclusion

We show that $W[1]$ -hardness precludes maintenance, and indeed given sufficient restriction on the stepwise computation no maintenance algorithm can guarantee any but the most trivial approximation ratios. However, as demonstrated with VERTEX COVER, some problems do admit fast, effective approximate maintenance algorithms. It is not clear whether there is a neat classification of these problems, however fixed-parameter tractability for the related parameterized (approximation) problem(s) is at least a necessary condition. The most likely situation is that there is some proper subset of the problems in FPT that admit a maintenance algorithm, as the structure of a maintenance algorithm seems to rely on some as yet undefined localizing property which is unlikely to be identical to the property of being fixed-parameter tractable. If indeed there is a coherent underlying property that allows maintenance, then perhaps it would also be possible to construct *maintenance-preserving* reductions (similar to the *incremental reductions* of [14]), which would allow a simpler and more robust demonstration of the possibility or impossibility of maintenance for a given problem.

References

1. G. Ausiello, V. Bonifaci, and B. Escoffier. Complexity and approximation in re-optimization. In S. B. Cooper and A. Sorbi, editors, *Computability in Context: Computation and Logic in the Real World*, pages 101–130. World Scientific, 2011.
2. B.-M. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of the Foundations of Computer Science*, 14(2):267–285, 2003.
3. A. Casteigts, S. Chaumette, and A. Ferreira. On the assumptions about network dynamics in distributed computing. *CoRR*, abs/1102.5529v4, 2012.
4. A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Deterministic computations in time-varying graphs: Broadcasting under unstructured mobility. In *IFIP TCS*, pages 111–124, 2010.
5. R. G. Downey and M. R. Fellows. *Parameterized complexity*. Springer, 1999.
6. R. G. Downey, M. R. Fellows, C. McCartin, and F. A. Rosamond. Parameterized approximation of dominating set problems. *Inf. Process. Lett.*, 109(1):68–70, 2008.
7. K. R. Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM*, pages 27–34, 2003.
8. P. Flocchini, B. Mans, and N. Santoro. Exploration of periodically varying graphs. In *ISAAC*, pages 534–543, 2009.
9. J. Flum and M. Grohe. *Parameterized complexity theory*. Springer, 2006.
10. J. Holm, K. de Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of the ACM*, 48(4):723–760, 2001.
11. S. Khot and O. Regev. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008.
12. D. Marx. Parameterized complexity and approximation algorithms. *Comput. J.*, 51(1):60–78, 2008.
13. L. Mathieson and S. Szeider. Editing graphs to satisfy degree constraints: A parameterized approach. *Journal of Computer and System Sciences*, 78(1):179–191, 2012.
14. P. B. Miltersen, S. Subramanian, J. S. Vitter, and R. Tamassia. Complexity models for incremental computation. *Theoretical Computer Science*, 130(1):203–236, 1994.
15. S. Patnaik and N. Immerman. Dyn-fo: A parallel, dynamic complexity class. *Journal of Computer and System Sciences*, 55(2):199–209, 1997.
16. E. Prieto. The method of extremal structure on the k-maximum cut problem. In M. D. Atkinson and F. K. H. A. Dehne, editors, *Theory of Computing 2005, Eleventh CATS 2005, Computing: The Australasian Theory Symposium, Newcastle, NSW, Australia, January/February 2005*, volume 41 of *CRPIT*, pages 119–126. Australian Computer Society, 2005.
17. L. Roditty and U. Zwick. A fully dynamic reachability algorithm for directed graphs with an almost linear update time. In L. Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 184–191. ACM, 2004.
18. S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23(3):555–565, 1976.
19. V. Weber and T. Schwentick. Dynamic complexity theory revisited. *Theory of Computing Systems*, 40(4):355–377, 2007.
20. Z. Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges. *IEEE Communications Surveys and Tutorials*, 8(1–4):24–37, 2006.

A Parameterized Approximation Complexity

In this section, we only provide the necessary definitions and properties of Parameterized Complexity and Parameterized Approximation Complexity. We refer the reader to [5,9] and [12] respectively for more details.

A.1 Basic Parameterized Complexity

Parameterized Complexity explores the complexity of combinatorial problems using parameters as independent measures of structure in addition to the overall size of the input. An instance (I, k) of a parameterized problem consists of the input I , corresponding to the input of a classical problem and an integer parameter k , a special part of the input independent from $|I|$.

A problem is *fixed-parameter tractable* (or in *FPT*) if there is an algorithm that solves each instance (I, k) of the problem in time bounded by $f(k) \cdot |I|^{O(1)}$ where f is a computable function dependent only on k .

Conversely hardness for any class in the W -hierarchy provides evidence that a problem is not fixed-parameter tractable. Hardness for such classes is typically established by *parameterized reduction*, the Parameterized Complexity reduction scheme where given an instance (I, k) of problem Π_1 an instance (I', k') of problem Π_2 is computed in time bounded by $f(k) \cdot |I|^{O(1)}$, with $k' \leq g(k)$ for some computable function g and (I, k) is a YES-instance if and only if (I', k') is a YES-instance.

A.2 Parameterized Approximation Complexity

The additional measure embodied in the parameter also provides an alternative possibility for approximation problems. Of particular interest is approximating the cost of the solution, where the parameter k is the desired cost of the solution. Given parameterized problem Π with an additional optimization objective (either min or max), the general cost minimization (maximization) parameterized problem for Π is:

$g(k)$ -APPROX- Π

Instance: A instance I of Π , an integer k .

Parameter: k .

Output: Either NO, asserting that there is no solution of size at most (at least) k for I , or a solution of size at most (at least) $g(k)$.

Of course for minimization problems the approximation is only interesting if $g(k) \geq k$, and vice versa for maximization. For specific functions $g(k)$ we obtain the following two interesting subcases:

c -ADD-APPROX- Π

Instance: A instance I of Π , an integer k .

Parameter: k .

Output: Either NO, asserting that there is no solution of size at most (at least) k for I , or a solution of size at most $k + c$ (at least $k - c$).

c-MULT-APPROX-*II*

Instance: A instance *I* of *II*, an integer *k*.

Parameter: *k*.

Output: Either NO, asserting that there is no solution of size at most (at least) *k* for *I*, or a solution of size at most *ck* (at least *k/c*).

Then $W[t]$ -hardness for any $t \geq 1$ for such an approximation problem gives evidence that there is no fixed-parameter tractable algorithm that can produce a solution of size bounded by $g(k)$ with k as the parameter.