Worst-case adjustment complexity for the problem of maximal independent set

Reading Group Algorithms

Antonis Skarlatos (12138293)

Abstract

In this paper we develop an algorithm which maintains a maximal independent set with $O(\sqrt{m})$ worst-case adjustment complexity. This bound is also tight based on a worst-case example by Assadi et al. [1]. We separate the set of vertices to heavy and light based on their degree. The algorithm maintains an invariant and a partial information for the light vertices.

Introduction

Consider a graph $G = \{V, E\}$, where $n = |V|$ and $m = |E|$ denote the number of vertices and edges of G respectively. A set of vertices $M \subseteq V$ is called an *independent set* if there is not an edge between two vertices of M. An independent set is *maximal (MIS)* if it is not a proper subset of any other independent set. The *state* of a vertex u indicates whether u belongs or not to M. The *adjustment complexity* of an update is the number of vertices whose state changes, that is, the number of vertices which either enter or leave M. The *worst-case adjustment complexity* of an algorithm is the worst possible adjustment complexity of an update. The worst-case adjustment complexity of the greedy algorithm which computes the MIS from scratch at each update is $O(n)$. Assadi et al. [1] presented an example such that any algorithm on it, would have $\Omega(n)$ worst-case adjustment complexity. In the next section we will develop an algorithm with $O(\sqrt{m})$ worst-case adjustment complexity, which improves the previous bound.

Worst-case adjustment complexity of MIS

In this section we will present an algorithm (Algorithm MIS) (source code in the Appendix) with $O(\sqrt{m})$ worst-case adjustment complexity. Let us give a high-level idea of the algorithm. We first separate the vertices into two sets, heavy and light, based on their degree. The change of the state of a heavy vertex could possibly affect its the whole neighborhood resulting in $\Omega(n)$ adjustments. To avoid this problem, the algorithm maintains an invariant such that any change of the state of heavy vertices does not affect the state of their light neighbors. To maintain the invariant though, a light vertex ignores its heavy neighbors with the danger of having a conflict while it is added to MIS. The conflict is resolved by the algorithm by applying the proper adjustments. The cost of the algorithm comes from the degree of a light vertex and from the size of the set of heavy vertices.

Consider the following partition of V into two sets H, L, defined as:

$$H = \{u \in V \mid \deg(u) > \sqrt{m}\}$$

$$L = \{u \in V \mid \deg(u) \leq \sqrt{m}\}$$

A vertex $u \in H$ is called *heavy* and a vertex $u \in L$ is called *light*. The trade-off comes from the fact that we would like the light vertices to have a small neighborhood and the heavy vertices to be as few as possible. For a graph G, it is true that $\sum_{u \in V} \deg(u) = 2m$. Thus it holds that $|H| \leq \frac{2m}{\sqrt{m}} = 2\sqrt{m}$.

The following lemma shows that the algorithm maintains an invariant regarding the light vertices. In particular, if a light vertex does not participate in the MIS, then it has definitely an other light neighbor in MIS. Hence changes in the state of heavy vertices do not affect the state of light vertices. As a result, there are not too many adjustments when the state of a heavy vertex changes.

Lemma 1. *Algorithm MIS maintains the following invariant: A light vertex $u \in L$ belongs to M if and only if it has not any light neighbor in M. In other words, for a vertex $u \in L$ it holds that: $u \in M \iff N(u) \cap L \cap M = \emptyset$.*

Proof. During the updates a heavy vertex does not modify the counter of a light neighbor. Hence the value of the counter of a light vertex $u \in L$ is equal to the number of its light neighbors in M, that is, equal to $|N(u) \cap L \cap M|$. A vertex u is inserted in M if and only if the corresponding value counter[u] is zero, thus a light vertex u is inserted in M if and only if u has not any light neighbor in M. Also for each two vertices connected by an edge, we always maintain only one of them in M with preference to light vertices. □

The next lemma proves the correctness of the algorithm.

Lemma 2. *The set M of Algorithm MIS is a maximal independent set.*

Proof. A vertex u is inserted in M if and only if the corresponding value counter[u] is zero. Heavy vertices have full information about their neighborhood, and so they do not cause any problem with M. Nevertheless, light vertices have only a partial information about their neighborhood. Specifically a light vertex has information only about its light neighborhood. Thus by inserting a light vertex into M could cause a conflict. This issue though is resolved by removing from M all the heavy neighbors. Therefore the set M is an independent set. It is also maximal because we always recursively insert a vertex whenever it is possible. □

The next theorem is the main result concerning the adjustment complexity of the algorithm.

Theorem 1. *The worst-case adjustment complexity of Algorithm MIS is $O(\sqrt{m})$.*

Proof. After an update, the set M might not be an MIS anymore. In order to remedy M, the state of a vertex u should change which in turn could affect the state of its neighbors. We call u the *main* vertex of the update. For each update there are two cases:

- The main vertex u is heavy: We claim that there are no adjustment in the light neighborhood of u, namely in the set $N(u) \cap L$. Vertex u keeps track of its light neighborhood in M, thus if there exists a vertex $w \in N(u) \cap L \cap M$, then u will not be inserted into M and will not affect the state of w. Moreover based on Lemma 1, if u leaves MIS then it also cannot affect the state of any light neighbor. Therefore u can only affect the state of its heavy neighbors, namely the set $N(u) \cap H$. In turn the affected heavy neighbors can only affect their heavy neighborhood, and so in total at most $|H| \leq 2\sqrt{m}$ vertices can be adjusted.
- The main vertex u is light: Vertex u can possibly affect the state of its whole neighborhood $N(u)$, where $|N(u)| \leq \sqrt{m}$. All light vertices keep track only of their light neighborhood in M. Thus there is the possibility that u or any of its light neighbors $N(u)$ will be inserted in M even though they have a heavy neighbor w in M. This would lead to the removal of any such w from M, which in turn could affect its own

neighborhood. From the previous case though, heavy vertices can only affect the state of their heavy neighborhood. Therefore in total at most $|u \cup N(u)| + |H| \leq 3\sqrt{m} + 1$ vertices can be adjusted.

In the fully dynamic setting the number of edges, namely the value of m, may change significantly after sufficient updates. As a result the sets H, L may not be consistent during the updates. Nevertheless, proving that during the updates it always holds that $|H| = O(\sqrt{m})$ and $\forall u \in L : |N(u)| = O(\sqrt{m})$, is sufficient to make the previous analysis work.

By increasing the value of m, a heavy vertex u may become light and this transformation could possibly affect the state of $N(u)$. In particular this could only happen if u is not in M and also has not any light neighbor in M. In this case, by converting u from heavy to light, u should enter M to maintain the invariant. This adjustment of u though, could lead to removals of heavy neighbors of u from M, which in turn could affect their own neighborhood. Based on the previous analysis, in total at most $O(|H|) = O(\sqrt{m})$ adjustments would take place, just for one vertex u. Such a vertex u moves from H to L, if before increasing m by one it holds that $\deg(u) = \sqrt{m+1}$. Let A be the set of those vertices, where the size of A can be as large as $\Omega(\sqrt{m})$ and so we cannot apply all the conversions from H to L at once, if we want to have $O(\sqrt{m})$ worst-case adjustment complexity. Instead at every iteration we will shift only one vertex from A to L. It is sufficient to show that a heavy vertex $u \in H \setminus A$ should move to L only after A has become empty, that is, only after $|A|+1$ updates, because then only the elements of A will not belong to the proper set. Observe that $|A| \leq 2\sqrt{m+1}$. For the first vertex $u \in H \setminus A$ which should move from H to L, it is true that $\deg(u) = \sqrt{m+1}+1$. The number of updates x until u should move to L, can be retrieved from the following relation $\sqrt{m+x} \geq \sqrt{m+1}+1$, thus $x \geq 2\sqrt{m+1}+2$. Therefore the number of updates is at least $|A|+1$, until the next batch of vertices which should shift from H to L arrives. Consequently, the size of H during this phase until H, L become consistent with their definition, can increase only by $|A|$. As a result it holds that $|H| = O(\sqrt{m'})$, by paying only $O(\sqrt{m'})$ adjustments per update, where m' is the new value of m after the updates.

$$L = \{1, \ldots, \sqrt{m}\}$$

$$H = \{\underbrace{\sqrt{m+1}, \ldots, \sqrt{m+1}}_{|A| \leq 2\sqrt{m+1}}, \sqrt{m+1}+1, \ldots, \Delta\}$$

Figure 1: Convert vertices from heavy to light

Similarly by decreasing the value of m, a light vertex u may become heavy and this transformation could possibly affect the state of $N(u)$. In particular this could only happen if u is in M and a light neighbor of u has not any other light neighbor in M besides u. In this case, by converting u from light to heavy, such a light neighbor $w \in N[u]$ should enter M to maintain the invariant. This adjustment of w though, could lead to removals of heavy neighbors of w from M (u will be one of them), which in turn could affect their own neighborhood. Based on the previous analysis, in total at most $O(|u \cup N(u)| + |H|) = O(\sqrt{m})$ adjustments would take place, just for one vertex u. Such a vertex u moves from L to H, if before decreasing m by one it holds that $\deg(u) = \sqrt{m}$. Let A be the set of those vertices, where the size of A can be as large as $\Omega(\sqrt{m})$ and so we cannot apply all the conversions from L to H at once, if we want to have $O(\sqrt{m})$ worst-case adjustment complexity. Instead at every iteration we will shift only one vertex from A to H. It is sufficient to show that a light vertex $u \in L \setminus A$ should move to H only after A has become empty, that is, only after $|A|+1$ updates, because then only the elements of A will not belong to the proper set. Observe that $|A| \leq 2\sqrt{m}$. For the first vertex $u \in L \setminus A$ which should move from L to H, it

is true that $\deg(u) = \sqrt{m} - 1$. The number of updates x until u should move to H, can be retrieved from the following relation $\sqrt{m-x} < \sqrt{m} - 1$, thus $x > 2\sqrt{m} - 1 \overset{x \text{ is integer}}{\iff} x \geq 2\sqrt{m}$. Therefore the number of updates is at least $|A|$, until the next batch of vertices which should shift from L to H arrives. By applying two shifts in an arbitrary update during this phase, the claim follows. Consequently, the degree of a vertex u in L during this phase until H, L become consistent with their definition, can be at most $\sqrt{m'} + 1$, where m' is the new value of m after the updates. As a result it holds that $\forall u \in L : |N(u)| = O(\sqrt{m'})$, by paying only $O(\sqrt{m'})$ extra adjustments per update.

$$H = \{\sqrt{m} + 1, \ldots, \Delta\}$$

$$L = \{1, \ldots, \sqrt{m} - 1, \underbrace{\sqrt{m}, \ldots, \sqrt{m}}_{|A| \leq 2\sqrt{m}}\}$$

Figure 2: Convert vertices from light to heavy

Moreover if an edge update affects the state of its endpoints, the corresponding shift takes place by paying only $O(\sqrt{m})$ extra adjustments. □

Therefore Algorithm MIS is correct with $O(\sqrt{m})$ worst-case adjustment complexity. Note that in the source code of Algorithm MIS, the process of updating the sets H, L has not been adjusted. However this is not hard to implement, by storing a list of vertices that have to be shifted and by applying the relevant updates.

Assadi et al. [1] display an example such that any algorithm that maintains an MIS would have $\Omega(n)$ worst-case adjustment complexity. Therefore as $\sqrt{m} \leq n$, there exists an example such that any algorithm that maintains an MIS would have $\Omega(\sqrt{m})$ worst-case adjustment complexity, and so the bound of our algorithm is tight.

Corollary 1. *The bound of Algorithm MIS is tight.*

References

[1] Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon. Fully dynamic maximal independent set with sublinear update time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, page 815–826, New York, NY, USA, 2018. Association for Computing Machinery.

[2] Manoj Gupta and Shahbaz Khan. Simple dynamic algorithms for maximal independent set, maximum flow and maximum matching. In *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 86–91. SIAM, 2021.

[3] Manoj Gupta and Shahbaz Khan. Simple dynamic algorithms for maximal independent set and other problems, 2018.

Appendix

Algorithm MIS

```
Update(u, oper) {
    queue = {}
    if (oper == "insert") { Insert u to M }
    else if (oper == "delete") { Delete u from M }
    // if u is heavy then it will inform only its heavy neighborhood
    // if u is light then it will inform its whole neighborhood
    if (u ∈ H) { A = H } else { A = N[u] }
    for (∀ w ∈ N[u] ∩ A) {
        if (oper == "insert") {
            counter[w] += 1
            // An insertion of a light vertex could lead to
            // a removal of a heavy neighbor
            if (u ∈ L and w ∈ H ∩ M) {
                queue.add(Update(w, "delete"))
            }
        } else if (oper == "delete") {
            counter[w] -= 1
            if (counter[w] == 0) { queue.add(Update(w, "insert")) }
        }
    }
    while(queue.nonempty()) {apply queue.front(), queue.pop()}
}

Delete_Edge(u, w) {
    Delete edge (u, w) from G
    // If u or w is in M, then assume w is certainly in M
    if (u ∈ M) { swap(u, w) }
    // A heavy vertex in M should not inform a light neighbor
    if (w ∈ M ∩ H and u ∈ L) { exit }
    if (w ∉ M) { exit }
    counter[u] -= 1
    if (counter[u] == 0) { Update(u, "insert") }
}

Insert_Edge(u, w) {
    Insert edge (u, w) to G
    // Prefer w to be in M and heavy
    if ((u ∈ M and w ∉ M) or (u ∈ M and w ∈ M and u ∈ H)) { swap(u, w) }
    // A heavy vertex in M should not inform a light neighbor not in M
    if (w ∈ M and not(w ∈ M ∩ H and u ∈ L \ M)) {
        if (u ∈ M) {
            counter[w] += 1
            Update(w, "delete")
        } else {
            counter[u] += 1
        }
    }
}
```

Figure 3: Algorithm with worst-case adjustment complexity $O(\sqrt{m})$