Average Sensitivity of Dynamic Programming

Soh Kumabe
The University of Tokyo
JST, PRESTO
soh.kumabe@mist.i.u-tokyo.ac.jp

Yuichi Yoshida
National Institute of Informatics
JST, PRESTO
yyoshida@nii.ac.jp

September 20, 2024

Abstract

When processing data with uncertainty, it is desirable that the output of the algorithm is stable against small perturbations in the input. Varma and Yoshida [SODA'21] recently formalized this idea and proposed the notion of average sensitivity of algorithms, which is roughly speaking, the average Hamming distance between solutions for the original input and that obtained by deleting one element from the input, where the average is taken over the deleted element.

In this work, we consider average sensitivity of algorithms for problems that can be solved by dynamic programming. We first present a $(1-\delta)$-approximation algorithm for finding a maximum weight chain (MWC) in a transitive directed acyclic graph with average sensitivity $O(\delta^{-1}\log^3 n)$, where n is the number of vertices in the graph. We then show algorithms with small average sensitivity for various dynamic programming problems by reducing them to the MWC problem while preserving average sensitivity, including the longest increasing subsequence problem, the interval scheduling problem, the longest common subsequence problem, the longest palindromic subsequence problem, the knapsack problem with integral weight, and the RNA folding problem. For the RNA folding problem, our reduction is highly nontrivial because a naive reduction generates an exponentially large graph, which only provides a trivial average sensitivity bound.

1 Introduction

Dynamic programming (DP) is a powerful framework used to solve a variety of optimization problems, including string problems, scheduling, and bioinformatics. DP is often used to solve problems like the *longest common subsequence problem* and the *longest increasing subsequence problem.*

In practice, it is important for algorithms to be *stable*, meaning the output should not significantly change with small changes in the input. For instance, if two people edit the same text file, one person's changes should not drastically alter the correspondence between the old and new files, allowing the second person to continue editing efficiently.

Varma and Yoshida [**VarmaYoshida**] introduced the concept of *average sensitivity* to formally study the stability of algorithms. The average sensitivity of a (randomized) algorithm ALG on an input set V of size n is defined as:

$$\frac{1}{n}\sum_{i\in V}\mathrm{EM}(\mathrm{ALG}(V),\mathrm{ALG}(V\setminus\{i\})),$$

where ALG(V) and ALG($V \setminus \{i\}$) are the distributions of the outputs on V and $V \setminus \{i\}$, respectively, and EM denotes the *Earth Mover's Distance* between two distributions:

$$\mathrm{EM}(\mathcal{X}_1, \mathcal{X}_2) := \min_{D} \mathbb{E}_{(\mathcal{X}_1,\mathcal{X}_2)\sim D}[|X_1 \Delta X_1|],$$

where D is a distribution over pairs of sets $\mathcal{X}_1$ and $\mathcal{X}_2$ with the same marginal distributions.

While it is natural to aim for stable-on-average algorithms, the typical DP algorithms tend to not be stable in this sense. This is because DP builds on the solutions to subproblems, and small changes in the input can propagate and amplify through the DP process.

1.1 Our Contributions

In this work, we design stable-on-average algorithms for various problems that can be solved by DP. To this end, we first design a stable-on-average algorithm for a problem called the maximum weight chain (MWC) problem, and we then reduce various problems to it. The details are follows.

1.1.1 Maximum Weight Chain

A directed acyclic subgraph (DAG) $G = (V, E)$ is called transitive if for any three vertices $v1, v2, v3 \epsilon V$ with $(v1, v2), (V2, V3) \epsilon E$ holds. In the MWC problem, we are given a weighted

Algorithm 1 : Naive DP for the MWC problem

```
Procedure DynamicProgramming($G = (V, E, w)$)
for $v \in V$ in their topological order do
    if $v$ has an incoming edge then
        $u^* \leftarrow \mathrm{argmax}_{u \in V, (u,v) \in E} w(\mathrm{DP}[u])$
        $\mathrm{DP}[v] \leftarrow \mathrm{DP}[u^*] \cup \{v\}$
    else
        $\mathrm{DP}[v] \leftarrow \{v\}$
return DP $[\mathrm{argmax}_{v \in V}(w(\mathrm{DP}[v]))]$ ;
```

transitive DAG $G = (V, E, w)$, where $w : V \to \mathbb{R}_+$ is a vertex weight function. The goal is to find a chain[1] P of vertices that maximizes the total weight, $\sum_{v \in P} w(v)$.
The MWC problem is a typical problem that can be solved by DP, as in Algorithm 1. Moreover, it serves as a target problem to which we can reduce various DP problems by regarding each vertex in G as a state of the source DP and each edge in G as the dependency between the states corresponding to the endpoints. Hence, if we have a stable-on-average algorithm for the MWC problem, we can automatically obtain stable-on-average algorithms for various other problems (unless the generated graph G is exponentially large).
When studying the average sensitivity of algorithms for the MWC problem, we use a slight extension of (1) that is convenient to show reductions from other DP problems. Let $G = (V, E, w)$ be a transitive DAG, and let $S_1, ..., S_n$ be antichains of G, that is, for any $i \in \{1, 2, ...n\}$, for any two vertices $u, v \in S_i$ do not form an edge in G. Now, the *average sensitivity of an algorithm* ALG *for the MWC problem on G with respect to* $S1, ..., S_n$ is defined by

[1]We save the word "path" here because we will use it later in the analysis of the ribonucleic acid (RNA) folding problem.

$$\frac{1}{n}\sum_{i=1}^{n} EM(ALG(V), ALG(VS_i)). \tag{1}$$

In the context of reducing some source DP to the MWC problem, n represents the number of elements in the instance of the source DP, and S_i corresponds to the set of the states of the source DP that would disappear if an element i is deleted from the instance. To emphasize this role of S_i, we call each S_i *potentially missing.*

In this work, we show that there is an approximation algorithm for the MWC problem with a nontrivially small average sensitivity:

Theorem 1.1. *For any $\delta > 0$, there is a polynomial-time randomized $(1-\delta)$-approximation algorithm for the MWC problem with the following property: Let $G = (V, E, w)$ be a transitive DAG and $S_1, \ldots, S_n$ be antichains such that each vertex in V appears in at least one and at most K of $S_1, \ldots, S_n$. Then, the average sensitivity of the algorithm on G with respect to $S_1, \ldots, S_n$ is $O\left(K\delta^{-1}\log^3|V|\right)$.*

We note that the linear dependency on δ^{-1} is necessary. Let $C > 2$ be a constant. Consider a disjoint union of the transitive closure of a chain with length $n/2$ and an antichain with size $n/2$. We set the weights of the vertices in the chain and the antichain as 1 and $\frac{1-C\delta}{2}n$, respectively. Then, any (randomized) $(1-\delta)-$approximation algorithm must output a subset of the chain with probability at least 1/2. However, if we delete random $2C\delta n$ vertices without replacement, then any $(1-\delta)-$approximation algorithm must output a vertex in the antichain with probability at least 1/2 because the weight of the chain becomes $\frac{1-2C\delta}{2}$ in expectation.Therefore, $2C\delta n$ times the average sensitivity is $\Omega(n)$ by the composition theorem of average sensitivity [**17**]; hence, the average sensitivity is $\Omega(\delta^{-1})$.

It is also natural to consider the worst-case sensitivity, which is obtained by replacing the average in (3) with the maximum over i's. However, there is no algorithm for the MWC problem with a reasonable approximation ratio and a small worst-case sensitivity. To see this, consider a transitive DAG consisting of the transitive closure of a long chain and an isolated vertex with a large weight. Although the isolated vertex forms the optimal solution in the original graph, we must use many vertices in the chain for the graph obtained by deleting the isolated vertex.

1.1.2 Applications

We can obtain stable-on-average algorithms for various DP problems by reducing them to the MWC problem (with constant K for measuring the average sensitivity). Here, we describe some representative examples.

Computing Differences Between Two Text Files. As discussed, when computing the differences between two text files, it is desirable to use stable-on-average algorithms. One of the most basic methods is to use the longest common subsequence. Another popular algorithm is the *patience dif* [**3**], which focuses more on lines whose copies appear only a small number of times in the files. This algorithm solves the longest increasing subsequence problem (see Section 3.1 for the definition) as a subroutine.

Because both of the longest common subsequence problem and the longest increasing subsequence problem can be solved by textbook DPs that can be formulated as an MWC, Theorem 1.1 immediately implies $(1-\delta)-$approximation algorithms with average sensitivity $O(\delta^{-1}\log^3 n)$ for these problems. See Sections 3.1 and 3.3 for more details.

Scheduling and Resource Allocation. The interval scheduling problem is one the most popular problems for scheduling tasks. In practice, tasks may be cancelled owing to several reasons such as the lack of participants or a bad weather. However, we do not want to drastically change the task schedule because that would incur huge cost. Hence, it is preferred to have a stable-on-average algorithm for the interval scheduling problem. Because this problem can be solved by a textbook DP that can be formulated as an MWC, Theorem 1.1 immediately implies $(1-\delta)-$ approximation algorithm with average sensitivity $O(\delta^{-1}\log^3 n)$ for this problem. See Section 3.2 for more details.

Another popular problem used to schedule tasks is the knapsack problem. If the costs of each task and the budget constraint, C, are integers, a textbook DP solves the knapsack problem in pseudo-polynomial time, where the pseudo-polynomial factor depends on C. Because this DP can be formulated as an MWC, Theorem 1.1 immediately implies a $(1-\delta)-$approximation algorithm with average sensitivity $O(\delta^{-1}\log^3 nC)$ for this problem. See Section 3.5 for more details.

Bioinformatics. For computational problems in bioinformatics, it is natural to assume that the input has some discrepancy with the true data because the process of observing a biological object injects errors. Therefore, if we want to obtain a useful output, then the algorithm used must be stable against errors; otherwise the output may not be significantly close to that of the true data. One approach to resolving this issue is to design algorithms with small average sensitivity.

Many problems of bioinformatics are solved by DP. For example, the interval scheduling problem is applied to the problem of determining protein structure from nuclear magnetic resonance (NMR) peak data [**1**, **19**] As another example, we note that DP have been used to determine the *secondary structure* of RNA, which represents how the RNA is physically folded. One popular formulation is the RNA *folding problem* proposed by Nussinov and Jacobson [**14**]. Besides its usefulness in bioinformatics, the RNA folding problem is theoretically interesting because it is a slight variant of *two- and one-dimensional (2D/1D) DP* [**6**], wherein a value of the DP array is given by the maximum over the sum of two values in the DP array. For example,

$$DP[i][j] = \max_{i \le k < j}(DP[i][k] + DP[k+1][j]),$$

which cannot be directly formulated as an MWC. To resolve this issue, we introduce a novel technique to formulate such a DP as an MWC using a quasi-polynomial number of vertices, and we show that there is a $(1-\delta)-$ approximation algorithm with average sensitivity $O(\delta^{-1}\log^7 nC)$. See Section 1.3 for a more detailed technical overview and Section 4 for the details of the algorithm and the analysis.

We note that the longest palindromic subsequence problem is a special case of the RNA folding problem. As opposed to the general RNA folding problem, the textbook DP algorithm for this can be directly formulated as an MWC, and hence Theorem 1.1 implies a $(1-\delta)$-approximation algorithm with average sensitivity $O(\delta^{-1}\log^3 nC)$ for this problem. See Sections 3.4 for more details.

1.2 Related Work

As mentioned, the notion of average sensitivity was recently proposed by Varma and Yoshida [**17**] They studied various graph problems, including the minimum spanning tree problem, minimum cut problem, and maximum matching problem and analyzed the average sensitivities of existing problems as well as developed new algorithms with small average sensitivities. Zhou and Yoshida [**20**] demonstrated a $(1-\epsilon)$-approximation algorithm for the maximum matching problem with sensitivity solely depending on ϵ, where sensitivity is defined as (3) with the average being replaced with the maximum over i. Peng and Yoshida analyzed the average sensitivity of spectral clustering [**15**], a popular method for graph clustering, and showed that it is stable-on-average if there is a relevant cluster structure in the input graph.

Kiirala et al. [**9**] investigated stable algorithms for the RNA folding problem. Their idea is to enumerate all bases that are paired in all optimal solutions and those that are never paired in any of the optimal solutions, then they construct the output using the enumerated bases.

1.3 Technical Overview

Maximum Weight Chain. Our algorithm is based on the divide-and-conquer method. For a vertex $v \in V$ in the input graph $G = (V, E, w)$, let V_{-v} (resp., V_{+v}) be the set of vertices v' such that there is a chain from v' to v (resp., from v to v'). Then, we pick a "middle" vertex $\bar{v}$ as a pivot and recurse on the two subgraphs induced by $V_{-\bar{v}}$ and $V_{+\bar{v}}$, respectively. The output of our algorithm is obtained by concatenating the result for $V_{-\bar{v}}$, the vertex $\bar{v}$, and the result for $V_{+\bar{v}}$ in this order.

We use the exponential mechanism [**12**] to select the pivot $\bar{v}$. Specifically, we sample a vertex $v \epsilon V$ as a pivot with probability proportional to $\exp(r(v)/c)$, where $r(v)$ is the maximum weight of a chain containing v, and c is an appropriately chosen constant.

We first discuss the approximation ratio. Because the maximum $r(v)$ over $v \epsilon V$ equals the optimal value of the original instance, in expectation, the optimal value does not decrease much by forcing v to be in the output. Indeed, for some appropriate choice of c, we can prove that the expected value of $r(v)$ is at least $1 - \epsilon$ times the optimal value of the original instance, where $\epsilon = O(\frac{\delta}{\log|V|})$ This means that, intuitively, one depth deeper in the recursion decreases the approximation ratio by ϵ. Hence, if the depth of recursion were to be $O(\log|V|)$, then the approximation ratio would be bounded by $1 - \delta$.

Generally, however, the depth of the recursion can go beyond $O(\log|V|)$.This is because the choice of $\bar{v}$ is not uniformly at random, and there is a chance that one of $V_{-\bar{v}}$ or V$_{+\bar{v}} hasalmostthesamesizeas$V$withh$
of vertices $v \epsilon V$ such that $|V_{-v}|$ and $|V_{+v}| \leq d$, where d is $|V|$ times some constant in (0, 1). We can prove that such a vertex set still has a vertex v such that $r(v)$ is equal to the optimal value of the original instance. With this modification, we can bound the depth of the recursion by $O(\log|V|)$ and obtain the approximation ratio of $1 - \delta$.

By contrast, the average sensitivity analysis is far more involved. The main part of our analysis is to prove that the distribution of the pivot $\bar{v}$ does not change much on average by deleting one of the potentially missing sets. Specifically, we bound the following average total variation distance:

$$\frac{1}{n}\sum_{i=1}^{n} \mathrm{TV}(ALG(V), ALG(V \backslash S_i)), \tag{2}$$

where $TV(\mathcal{X}_1, \mathcal{X}_2)$ represents the total variation distance of two output distributions $\mathcal{X}_1$ and $\mathcal{X}_2$. Let us assume for now that the average total variation distance is small. To bound the average sensitivity of our algorithm, for each $i \ \epsilon \ \{1, 2, \ldots, n\}$, we transport probability mass of $ALG(V)$ corresponding to a particular choice of the pivot $\bar{v}$ to that of $ALG(V \backslash S_i)$ corresponding to the same $\bar{v}$ as far as possible[2] In this way, we can transport $1 TV(ALG(V), ALG(V \backslash S_i))$ amount of probability mass for each i. For the probability mass transported this way, we recursively transport probability mass from $ALG(V_{-\bar{v}})$ to $ALG(V_{-\bar{v}} \backslash S_i)$ and from $ALG(V_{+\bar{v}})$ to $ALG(V_{+\bar{v}} \backslash S_i)$, and then

[2]When we bound the earth mover's distance from above, we can use any joint distribution $\mathcal{D}$ because we take the minimum over all possible joint distributions in (2). In our analysis, we often construct $\mathcal{D}$ by specifying how we transport probability mass from one distribution to the other.

apply the same analysis. The remaining probability mass of $TV(ALG(V), ALG(V\backslash S_i))$ is transported arbitrarily. Its contribution to the average sensitivity can be bounded by $TV(ALG(V), ALG(V\backslash S_i))\cdot n$, which is small.

We now explain how we bound (4). An important observation is that when we delete a potentially missing set uniformly at random, the value $r(v)$ decreases by at most $\frac{r(v)}{n}$ in expectation for every $v \epsilon V$. Then one may think that if the factor c is chosen appropriately, the decrease of the probability to select a particular v as a pivot is small in expectation. However, this idea does not work. The main reason for this is that we sample a pivot from U_d, not V . When $|V|$ decreases a lot by deleting a potentially missing set, a large number of vertices may join U_d, and the probability of choosing a vertex as a pivot may drastically change.

To resolve this issue, we sample the threshold d from an interval, e.g., $[\frac{1}{2}|V|, \frac{3}{4}|V|]$. Then, we analyze the average sensitivity by transporting the probability mass of $ALG(V)$ corresponding to a particular choice of the threshold to that of $ALG(V\backslash S_i)$ corresponding to the same threshold. Note that we can do so, except for the probability mass that the threshold is in $ALG(V)$ is in $[\frac{3}{4}|V\backslash S_i|, \frac{3}{4}|V|]$. However, the average of this mass over i is small and does not contribute much to the average sensitivity. Similar issues arise when we choose the scaling factor c and the value ϵ because they depend on $|V|$. We can also resolve them by sampling these values from some intervals instead of fixing these values uniquely.

RNA Folding. Here, we describe the intuition behind our reduction from the RNA folding problem to the MWC problem. Our reduction is inspired by a pseudo-polynomial time algorithm for solving constrained knapsack problems on trees [**11**], in which they reduced the dependency of the time complexity on the weight limit from quadratic to linear. Before getting into details, we formally define the RNA folding problem.

Problem 1 (RNA folding [**14**]). *Let A be a string of length n over the alphabet $\sum$. Let $R \subseteq \sum \times \sum$ be a binary relation. Two pairs of indices (l, r) and $(l^{'}, r^{'})$ are* pseudoknot *if exactly one of $l^{'}$ and $r^{'}$ is located between l and r, exclusively. The output is a set of pairs of indices $\{(l_1, r_1), \dots, (l_t, r_t)\}$ such that any two of $l_1, r_1, \dots, l_t, r_t$ are distinct. The goal is to maximize t subject to $l_i < r_i$, $(A_{l_i}, A_{r_i}) \ \epsilon \ \mathcal{R}$ for all $i \ \epsilon \ \{1, \dots, t\}$, and no two different pairs in the output form a pseudoknot.*

In this work, the average sensitivity of an algorithm ALG for the RNA folding problem is defined as $\frac{1}{n}\sum_{i=1}^{n} d_{EM}(ALG(A), ALG(A^i))$, where for $i \ \epsilon \ \{1, 2, \dots, n\}$, $A^i = (a_1, \dots, a_{i-1}, a_{i+1}, \dots a_n)$ is the sequence obtained from A by dropping a_i and the distance between two solutions used in the earth mover's distance is the size of their symmetric difference.

Let A be an instance of the RNA folding. We want to construct a transitive DAG G such that a chain in G corresponds to a solution for A of the same weight and vice versa.

First, we observe that a feasible solution $X = \{(l_1, r_1), \dots, (l_t, r_t)\}$ for an instance A of the RNA folding problem defines the transitive closure of a forest T_X as follows. We introduce a vertex in T_X for each pair in X. Then, we add an edge from

$(l, r) \ \epsilon \ X$ to another $(l^{'}, r^{'}) \ \epsilon \ X$ whenever $[l^{'}, r^{'}] \subseteq [l, r]$. The resulting graph is the transitive closure of a forest because the feasible solution does not have a pseudoknot. For the sake of analysis, we introduce a root vertex in T_X corresponding to a pair $(0, n+1)$ and regard T_X as a rooted tree.

For a feasible solution X and $(l^{'}, r^{'}) \ \epsilon \ X$, let $P_{X,i}$ be a path in T_X from the root to the vertex $(l^{'}, r^{'})$. Let us consider a graph $G^{'}$ whose vertex set is the set of all possible paths $\{P_{X,i}\}_{X,i}$. We introduce only one vertex even if the same path arises from different feasible solutions. For each feasible solution X and a pair of base pairs $(l_i, r_i), (l^{'}, r^{'})$ in X, we introduce an edge from $P_{X,i}$ to $P_{X,i^{'}}$ if in some fixed pre-order transversal of T_x, (l_i, r_i) appears earlier than $(l^{'}, r^{'})$. We can then show that the resulting graph $G^{'}$ becomes acyclic and transitive if the pre-order transversals are consistent among X's in a certain sense, and each chain in G corresponds to a feasible solution for the original instance and vice versa.

An issue here is that the size of $G^{'}$ is exponentially large, and thus Theorem 1.1 applied on G gives a polynomial bound on the average sensitivity, which is trivial. To resolve this issue, we consider the *heavy-light decomposition* of T_X. An edge (u, v) in T_X, where v is a child of u, is heavy if the size of the subtree rooted at v is the maximum among those of all children of u. Otherwise, it is light. Ties are broken arbitrarily so that each non-leaf vertex in T_X has exactly one heavy child. Then, we construct a graph G as follows. For each feasible solution X and $(l_i, r_i) \in X$, let $Q_{X,i}$ be the list of all light edges in the path $P_{X,i}$. The vertex set of G is the set of all possible lists $\{Q_{X,i}\}_{X,i}$. We introduce only one vertex even if the same list arises from different feasible solutions. For each feasible solution X and a pair of base pairs $(l_i, r_i), (l_{i'}, r_{i'})$ in X, we introduce an edge from $Q_{X,i}$ to $Q_{X,i'}$ if in some fixed pre-order transversal of $T_X, (l_i, r_i)$ appears earlier than $(l_{i'}, r_{i'})$. An important observation here is that because there are at most $\log n$ light edges on a path in T_X and hence in $Q_{X,i}$, the size of $V(G)$ is bounded by $n^(O(\log n))$. Therefore, by applying Theorem 1.1 on G, we obtain a polylogarithmic average sensitivity bound.

1.4 Organization

The rest of this paper is organized as follows. In Section 2, we introduce our algorithm for the MWC problem and analyze its approximation ratio and average sensitivity. In Section 3, we discuss some DP problems for which we can directly obtain stable-on-average algorithms by reducing to the MWC problem. Finally, in Section 4, we show a stable-on-average algorithm for the RNA folding problem.

2 Stable-on-average Algorithm for the Maximum Weight Chain Problem

In this section, we prove Theorem 1.1. We describe our algorithm and show its basic properties in Section 2.1. We analyze the approximation ratio and average sensitivity in Sections 2.2 and 2.3, respectively. The proof of a key technical lemma (Lemma 2.6) in the analysis of average sensitivity is provided in Section 2.4.

2.1 Algorithm Description and Basic Properties

Let $G = (V, E, w)$ be a transitive DAG with a vertex weight function $w : V \to \mathbb{R}_+$. For a vertex set $U \subseteq V$ and a vertex $v \in U$, let $U_{-v} \subseteq V$ (resp., $U_{+v} \subseteq V$) be the set of vertices u such that there is an edge from u to v (resp., from v to u). Note that owing to the transitivity of G, if there is a chain from u to v, then there is an edge from u to v, and hence $u \in U_{-v}$ holds. Let $S_1, \ldots, S_n$ be potentially missing antichains of G such that each vertex in V is contained in at least one and at most K of them. Because a chain cannot have two or more vertices from the same S_i , the size of any chain in G is at most n. Let $U \subseteq V$ be a vertex set. Then, let $w(U) = \sum_{v \in U} w(v), G[U]$ be the subgraph of G induced by U, and $\mathrm{opt}(U)$ be the maximum weight of a chain in $G[U]$.

Our algorithm is given in Algorithm 2. Given a vertex set U, we select a vertex vU, which we call a *pivot*, in a nearly optimal chain with respect to $G[U]$. Then we recursively apply the algorithm on U_{-v} and U_{+v}. To bound the depth of the recursion, we select a vertex v from U_d (defined at Line 8) so that both $|U_{-v}|$ and $|U_{+v}|$ are of at most a constant, say $\frac{3}{4}$, fraction of $|U|$. Clearly, the running time is polynomial.

The following lemma ensures that an optimal chain has a vertex in U_d for any $d \geq \frac{1}{2}|U|$.

Lemma 2.1. *For any $U \subseteq V$ and $d \geq \frac{1}{2}|U|$, there is a vertex $v \epsilon U_d$ with $r(v) = opt(U)$, where $r(v)$ is as defined at Line 7 of Algorithm 2. In particular,* $\max_{v \epsilon U_d} r(v) = opt(U)$.

Proof. Let$P = (v_1, \dots, v_k)$ be a maximal chain that attains $opt(U)$. Let i be the last index with $|U_{-v_i}| \leq \frac{|U|}{2}$. We prove $|U_{+v_i}| \leq \frac{|U|}{2}$.

Algorithm 2 : Stable-on-average algorithm for the maximum weight chain problem

```
Procedure REC($U, \epsilon$)
 1: Sample $c$ uniformly from $\left[\frac{\epsilon opt(U)}{\log(|U|\epsilon^{-1})}, 2 \cdot \frac{\epsilon opt(U)}{\log(|U|\epsilon^{-1})}\right]$;
 2: Sample $d$ uniformly from $\left[\frac{1}{2}|U|, \frac{3}{4}|U|\right]$;
 3: if $U = \emptyset$ then
 4:    return $\emptyset$;
 5: end if
 6: for each $v \in U$ do
 7:    Let $r(v)$ be the maximum weight of a chain that includes $v$;
 8: end for
 9: Let $U_d$ be the set of vertices $v \in U$ with $\max(|U_{-v}|, |U_{+v}|) \leq d$;
10: Sample $\bar{v} \in U_d$ with probability proportional to $\exp(r(\bar{v})/c)$;
11: return REC($U_{-\bar{v}}, \epsilon$) $\cup \{\bar{v}\} \cup$ REC($U_{+\bar{v}}, \epsilon$);
Procedure MWC($G = (V, E, w), \delta$)
12: Sample $\epsilon^{-1}$ uniformly from $\left[17\delta^{-1}\log(|V|), 34\delta^{-1}\log(|V|)\right]$;
13: return REC($V, \epsilon$); =0
```