Quantum Information and Computation, Vol. 0, No. 0 (2003) 000–000
© Rinton Press

QUANTUM AND CLASSICAL ALGORITHMS FOR APPROXIMATE SUBMODULAR FUNCTION MINIMIZATION

YASSINE HAMOUDI[a], PATRICK REBENTROST[b], ANSIS ROSMANIS[c], and MIKLOS SANTHA[d]

Received (received date)
Revised (revised date)

Submodular functions are set functions mapping every subset of some ground set of size n into the real numbers and satisfying the diminishing returns property. Submodular minimization is an important field in discrete optimization theory due to its relevance for various branches of mathematics, computer science and economics. The currently fastest strongly polynomial algorithm for exact minimization [1] runs in time $\widetilde{O}(n^3 \cdot \mathrm{EO} + n^4)$ where EO denotes the cost to evaluate the function on any set. For functions with range $[-1, 1]$, the best ϵ-additive approximation algorithm [2] runs in time $\widetilde{O}(n^{5/3}/\epsilon^2 \cdot \mathrm{EO})$.

In this paper we present a classical and a quantum algorithm for approximate submodular minimization. Our classical result improves on the algorithm of [2] and runs in time $\widetilde{O}(n^{3/2}/\epsilon^2 \cdot \mathrm{EO})$. Our quantum algorithm is, up to our knowledge, the first attempt to use quantum computing for submodular optimization. The algorithm runs in time $\widetilde{O}(n^{5/4}/\epsilon^{5/2} \cdot \log(1/\epsilon) \cdot \mathrm{EO})$. The main ingredient of the quantum result is a new method for sampling with high probability T independent elements from any discrete probability distribution of support size n in time $O(\sqrt{Tn})$. Previous quantum algorithms for this problem were of complexity $O(T\sqrt{n})$.

Keywords: Submodular functions, approximate minimization, quantum algorithms, subgradient descent
Communicated by: to be filled by the Editorial

1 Introduction

1.1 Submodular Minimization

A submodular function F is a function mapping every subset of some finite set V of size n into the real numbers and satisfying the diminishing returns property: for every $A \subseteq B \subseteq V$ and for every $i \notin B$, the inequality $F(A \cup \{i\}) - F(A) \geq F(B \cup \{i\}) - F(B)$ holds. In words, given two sets where one of them contains the other, adding a new item to the smaller set increases the function value at least as much as adding that element to the bigger set. Many classical functions in mathematics, computer science and economics are submodular, the most prominent examples include entropy functions, cut capacity functions, matroid rank functions and utility functions. Applications of submodular functions, or slight variants of them, occur in areas as far reaching as machine learning [3, 4, 5], operations research [6, 7], electrical networks [8], computer vision [9], pattern analysis [10] and speech analysis [11].

[a] Université de Paris, IRIF, CNRS, F-75013 Paris, France.
[b] Centre for Quantum Technologies, National University of Singapore, Singapore 117543.
[c] Centre for Quantum Technologies, National University of Singapore, Singapore 117543.
[d] Université de Paris, IRIF, CNRS, F-75013 Paris, France; and Centre for Quantum Technologies and MajuLab UMI 3654, National University of Singapore, Singapore 117543.

Submodular functions show analogies both with concavity and convexity. The diminishing returns property makes them akin to concave functions, but they have algorithmic properties similar to convex functions. In particular, while it follows from the NP-hardness of maximum cut that submodular maximization is NP-hard, submodular minimization can be solved in polynomial time, in fact even in strongly polynomial time. The link between submodular functions and convex analysis is made explicit through the Lovász extension [12]. There are various approaches to solve submodular minimization. The foundational work of Grötschel, Lovász and Schrijver [13] gave the first polynomial time algorithm using the ellipsoid method. The first pseudo-polynomial algorithm using a combinatorial method appeared in the influential paper of Cunningham [14]. In a later work Grötschel, Lovász and Schrijver [15] were the first to design a strongly polynomial time algorithm, and the first strongly polynomial time combinatorial algorithms were given by Schrijver [16] and by Iwata, Fleischer and Fujishige [17]. Many of these works assume an access to an evaluation oracle for the function F, where the time of a query is denoted by EO.

The current fastest submodular minimization algorithm is by Lee, Sidford and Wong [1]. Their weakly polynomial algorithm runs in time $\widetilde{O}(n^2 \log M \cdot \mathrm{EO} + n^3 \log^{O(1)} M)$ and their strongly polynomial algorithm runs in time $\widetilde{O}(n^3 \cdot \mathrm{EO} + n^4)$, where M is an upper bound on the integer valued function and the notation $\widetilde{O}()$ hides polylogarithmic factors in n. Both algorithms apply a new cutting plane method given in the same paper and use the Lovász extension. Our work is most closely related to the recent paper of Chakrabarty et al. [2] who gave an $\widetilde{O}(nM^3 \cdot \mathrm{EO})$ algorithm in the setting of [1], and an ϵ-additive approximation algorithm that runs in time $\widetilde{O}(n^{5/3}/\epsilon^2 \cdot \mathrm{EO})$ for real valued submodular functions with range $[-1, 1]$. These algorithms were the first to run in subquadratic time in n, by going beyond the direct use of the subgradients of the Lovász extension. Indeed, it is proven in [2] that any algorithm accessing only subgradients of the Lovász extension has to make $\Omega(n^2)$ queries. Such algorithms include [1] and the Fujishige-Wolfe algorithm [18, 19]. Subquadratic approximate algorithms (such as in [2] or our work) can potentially lead to insights to the exact case. They are also more practical when the scaling in n is more important than whether or not the algorithm is exact.

1.2 Quantum Algorithms for Optimization

A quite successful recent trend in quantum computing is to design fast quantum algorithms for various optimization and machine learning problems. At a high level, these algorithms are constructed in various subtly different input/output models [20, 21, 22, 23]. In the model that we are working with in this paper, the input is given by an oracle that can be accessed in quantum superposition, and the output is classical. The algorithms in this model are often hybrid, that is partly of classical and partly of quantum nature, and designed in a modular way so that the quantum part of the algorithm can be treated as a separate building block. In fact, a standard feature of these algorithms is that they make a quantum improvement on some part of the (best) available classical algorithm, but they keep its overall structure intact. In most cases the quantum versus classical speed-up is at most polynomial, usually at most quadratic. While we expect the quantum algorithm to deliver some speed-up at least in one of the input parameters, sometimes it might be worse than the best classical algorithm in some other parameters.

We mention here some of the fastest optimization algorithms in the quantum oracle model. In this paragraph we use the notation $O^*()$ to hide polylogarithmic factors in any of the arguments. For solving an SDP with m constraints involving $n \times n$ matrices, van Apeldoorn and Gilyén [23] gave an algorithm that runs in time $O^*\big((\sqrt{m} + \frac{\sqrt{n}}{\gamma})s/\gamma^4\big)$ where s is the row-sparsity of the input matrices and $\gamma = \epsilon/Rr$ is the additive error ϵ of the algorithm scaled down with the upper bounds R and r on the respective sizes of the primal and dual solutions. This result builds on the classical Arora-Kale framework [24] that runs in time $O^*(mns/\gamma^4 + ns/\gamma^7)$, which was first quantized by Brandão and Svore [25]. In [26] Li, Chakrabarti and Wu gave an $O^*(\sqrt{n}/\epsilon^4 + \sqrt{d}/\epsilon^8)$ time quantum algorithm for the classification of n data points in dimension d with margin ϵ. Their design is the quantization of the work of Clarkson, Hazan and Woodruff [27] that runs in time $O^*((n+d)/\epsilon^2)$. The same paper contains similar quadratic quantum improvements over the classical constructions of [27] for kernel based classification, minimum enclosing ball and ℓ^2-margin SVM, as well as an $O^*(\sqrt{n}/\epsilon^4)$ time algorithm for zero-sum games with $n \times n$ payoff matrices. A similar result for zero-sum games, but with a better dependence on the error parameter, was obtained by van Apeldoorn and Gilyén [28] whose algorithm runs in time $O^*(\sqrt{n}/\epsilon^3)$. Both quantum algorithms for zero-sum games are based on the classical work of Grigoriadis and Khachiyan [29] whose complexity is $O^*(n/\epsilon^2)$. Finally, there is a series of quantum algorithms [30, 31, 32, 33] for fast gradient computation, which are typically combined with classical first order methods such as gradient descent.

1.3 Previous Work

The previous work on approximate submodular minimization [34, 3, 2] is based on the subgradient descent method applied to the Lovász extension. We review these results below, as it will help to present our contributions in the next section. From now on, we restrict our attention to submodular functions F with range $[-1, 1]$ and we seek for a set $\bar{S}$ such that $F(\bar{S}) \leq \min_S F(S) + \epsilon$.

Subgradient descent in [34]. Submodular minimization can be translated into a convex optimization problem by considering the so-called Lovász extension f. This makes it possible to apply standard gradient algorithms. Since f is not differentiable, one can rely on the subgradient descent method that computes a sequence of iterates $x^{(t)}$ converging to a minimum of f. At each step, the next iterate $x^{(t+1)}$ is obtained by moving into the negative direction of a subgradient $g^{(t)}$ at $x^{(t)}$. In the case of submodular functions, there exists a natural choice for $g^{(t)}$, sometimes called the Lovász subgradient, that requires $O(n/\epsilon^2)$ steps to converge to an ϵ-approximate of the minimum. Since the Lovász subgradient can be computed in time $O(n \cdot \mathrm{EO} + n \log n)$, the complexity of this approach is $\widetilde{O}(n^2/\epsilon^2 \cdot \mathrm{EO})$ [34]. The question arises if it is possible to find algorithms that scale better than n^2, i.e. that are subquadratic in the dimension.

Stochastic subgradient descent in [2]. In the above method, the subgradient $g^{(t)}$ can equally be replaced with a stochastic subgradient, that is a low-variance estimate $\widetilde{g}^{(t)}$ satisfying $\mathbb{E}[\widetilde{g}^{(t)} \mid x^{(t)}] = g^{(t)}$. One possible choice for $\widetilde{g}^{(t)}$ is the *subgradient direct estimate* $\widehat{g}^{(t)}$ defined as $\widehat{g}^{(t)} = \|g^{(t)}\|_1 \operatorname{sgn}(g_i^{(t)}) \cdot \vec{1}_i$ where $i \in [n]$ is sampled with probability $|g_i^{(t)}|/\|g^{(t)}\|_1$. The ℓ_1-norm of the Lovász subgradient being small, this is a low-variance 1-sparse estimate of

$g^{(t)}$. However, it is unknown how to sample $\widehat{g}^{(t)}$ faster than $O(n \cdot \mathrm{EO} + n \log n)$. Thus, using $\widehat{g}^{(t)}$ at each step of the descent would not be more efficient than using the actual subgradient $g^{(t)}$. Instead, the approach suggested in [2] is to use $\widetilde{g}^{(t)} = \widehat{g}^{(t)}$ only once every $T = n^{1/3}$ steps, and in between to use $\widetilde{g}^{(t)} = \widetilde{g}^{(t-1)} + \widetilde{d}^{(t)}$ where $\widetilde{d}^{(t)}$ is an estimate of the subgradient difference $d^{(t)} = g^{(t)} - g^{(t-1)}$. The crucial result in [2] is to show how to construct $\widetilde{d}^{(t)}$ in time only proportional to the sparsity of $x^{(t)} - x^{(t-1)}$. This process has to be reset every T steps since the variance and the sparsity of $\widetilde{g}^{(t)}$ increase over time (and therefore the sparsity of $x^{(t)} - x^{(t-1)}$ too). The amortized cost per step for constructing $\widetilde{g}^{(t)}$ is $\widetilde{O}(n^{2/3} \cdot \mathrm{EO})$, leading to an $\widetilde{O}(n^{5/3}/\epsilon^2 \cdot \mathrm{EO})$ algorithm.

It seems to us that there is a slight error in the construction of [2] because the estimate $\widetilde{g}^{(t)} = \widetilde{g}^{(t-1)} + \widetilde{d}^{(t)}$ is not a valid stochastic subgradient, that is $\mathbb{E}[\widetilde{g}^{(t)} \mid x^{(t)}] \neq g^{(t)}$. Indeed, even if $\widetilde{g}^{(t-1)}$ is an unbiased estimate of $g^{(t-1)}$ conditioned on $x^{(t-1)}$, in general that is not true conditioned on $x^{(t)}$ (see a counterexample in Appendix 6), thus $\mathbb{E}[\widetilde{g}^{(t)} \mid x^{(t)}] = \mathbb{E}[\widetilde{g}^{(t-1)} \mid x^{(t)}] + g^{(t)} - g^{(t-1)} \neq g^{(t)}$. Nonetheless, this problem can be easily solved by sampling a second estimate $\widetilde{\widetilde{g}}^{(t-1)}$ of $g^{(t-1)}$ such that, when conditioned on $x^{(t-1)}$, it becomes independent of $\widetilde{g}^{(t-1)}$. Then $\widetilde{g}^{(t)}$ is redefined as $\widetilde{g}^{(t)} = \widetilde{\widetilde{g}}^{(t-1)} + \widetilde{d}^{(t)}$. In this case, $x^{(t)}$ does not convey any information about $\widetilde{\widetilde{g}}^{(t-1)}$ when conditioned on $x^{(t-1)}$, implying that $\mathbb{E}[\widetilde{\widetilde{g}}^{(t-1)} \mid x^{(t)}] = g^{(t-1)}$. However, in order to make $\widetilde{\widetilde{g}}^{(t-1)}$ independent of $\widetilde{g}^{(t-1)}$, *all the previous* estimates involved in the computation of $\widetilde{g}^{(t-1)}$ have to be resampled. Since the construction of $\widetilde{g}^{(t)}$ was decomposed into batches of T steps, it means that there are between 1 and T estimates to resample at each step. Nevertheless, a straightforward analysis shows that asymptotically the time complexity remains as stated in [2].

1.4 Our Contributions

Our method also consists of minimizing the Lovász extension of the submodular function under consideration by using the stochastic subgradient descent algorithm. We differ from [34, 2] by constructing a new subgradient oracle that is faster to evaluate. In the quantum model, our further speed-up is based on two new results that might be of independent interest. One is a simple proof of robustness for the (classical) stochastic subgradient descent method when the subgradient oracle has some biased noise. The other one is a new quantum algorithm for sampling multiple independent elements from discrete probability distributions. These results are detailed below.

Classical algorithm. Similarly to [2], we construct our subgradient oracle $\widetilde{g}^{(t)}$ by combining two kinds of estimates (Section 4). Our construction is reset every $T = n^{1/2}$ steps, which turns out to be the optimal resetting time in our case. We explain how to compute the first T terms $\widetilde{g}^{(0)}, \ldots, \widetilde{g}^{(T-1)}$. First, we obtain T independent samples $\widehat{g}^{(0,0)}, \cdots, \widehat{g}^{(0,T-1)}$ from the subgradient direct estimate at $x^{(0)}$. Using a standard sampling method (Lemma 1), this can be done in time $\widetilde{O}(n \cdot \mathrm{EO} + T)$ (Proposition 5). Then, the first subgradient estimate is chosen to be $\widetilde{g}^{(0)} = \widehat{g}^{(0,0)}$, and the other ones are obtained at step t by combining $\widehat{g}^{(0,t)}$ with an estimate $\widetilde{d}^{(t)}$ of the Lovász subgradient difference $d^{(t)} = g^{(t)} - g^{(0)}$, that is $\widetilde{g}^{(t)} = \widehat{g}^{(0,t)} + \widetilde{d}^{(t)}$. Notice that the difference is not taken between two consecutive iterates $x^{(t-1)}$ and $x^{(t)}$ as in [2], but between the first iterate $x^{(0)}$ and the current one $x^{(t)}$. This has the advantage of keeping the variance under control since we add up two terms instead of $t + 1$. Moreover, the sparsity increases only linearly, instead of quadratically, in t. Our procedure for constructing $\widetilde{d}^{(t)}$ is directly adapted from [2], with time complexity $\widetilde{O}(t \cdot \mathrm{EO})$ (Proposition 6). Consequently, the first T estimates are obtained in time $\widetilde{O}\big((n \cdot \mathrm{EO} + T) + \sum_{t=1}^{T-1} t \cdot \mathrm{EO}\big) = \widetilde{O}(n \cdot \mathrm{EO})$. Since the $O(n/\epsilon^2)$ steps of the subgradient descent are split into $O(\sqrt{n}/\epsilon^2)$ batches of length T, it follows that the total time complexity is $\widetilde{O}(n^{3/2}/\epsilon^2 \cdot \mathrm{EO})$.

Statement of Theorem 4 *There is a classical algorithm that, given a submodular function $F : 2^V \to [-1,1]$ and $\epsilon > 0$, computes a set $\bar{S}$ such that $\mathbb{E}[F(\bar{S})] \leq \min_{S \subseteq V} F(S) + \epsilon$ in time $\widetilde{O}(n^{3/2}/\epsilon^2 \cdot \mathrm{EO})$.*

Quantum algorithm. We first note that there is a simple $\widetilde{O}(n^{3/2}/\epsilon^3 \cdot \mathrm{EO})$ quantum algorithm using only the subgradient direct estimate $\widehat{g}^{(t)}$. The latter was defined as $\widehat{g}^{(t)} = \|g^{(t)}\|_1 \operatorname{sgn}(g_i^{(t)}) \cdot \vec{1}_i$ where $i \in [n]$ is sampled from the probability distribution $\big(|g_1^{(t)}|/\|g^{(t)}\|_1, \ldots, |g_n^{(t)}|/\|g^{(t)}\|_1\big)$. It is a standard result that one sample from any discrete probability distribution $(p_1, \ldots, p_n)$ (given as an evaluation oracle) can be obtained in time $O(\sqrt{n \cdot \max_i p_i} \cdot \mathrm{EO}) = O(\sqrt{n} \cdot \mathrm{EO})$ by quantum state preparation of $\sum_{i \in [n]} \sqrt{p_i} |i\rangle$ (Lemma 2). Moreover, the ℓ_1-norm of any n-coordinates vector can be estimated with accuracy ϵ in time $O(\sqrt{n}/\epsilon \cdot \mathrm{EO})$ using the Amplitude Estimation algorithm (Lemma 3). A straightforward combination of these two results leads to an ϵ-biased estimate $\widehat{g}_\epsilon^{(t)}$, satisfying $\|\mathbb{E}[\widehat{g}_\epsilon^{(t)} \mid x^{(t)}] - g^{(t)}\|_1 \leq \epsilon$, that can be computed in time $\widetilde{O}(\sqrt{n}/\epsilon \cdot \mathrm{EO})$. This does not meet the usual requirement of an unbiased estimate for the stochastic subgradient descent method. However, we prove that the latter is robust to such a noise (Proposition 3). This leads to an $\widetilde{O}(n^{3/2}/\epsilon^3 \cdot \mathrm{EO})$ algorithm for approximate submodular minimization.

We now describe our quantum algorithm achieving time complexity $\widetilde{O}(n^{5/4}/\epsilon^{5/2} \cdot \log(1/\epsilon) \cdot \mathrm{EO})$, based on our enhanced classical algorithm. Similarly to the simple result described above, we accelerate the construction of the subgradient estimate $\widetilde{g}^{(t)}$ using quantum sampling. A first attempt would be to apply the quantum state preparation method to sample each estimate individually. However, the computation of the first T estimates of $g^{(0)}$, for instance, would incur a cost of $\widetilde{O}(T\sqrt{n}/\epsilon \cdot \mathrm{EO})$, which is worse than classically when $T = n^{1/2}$ (we could change the value of T but that does not improve the overall complexity). We overcome this issue by using a new quantum multi-sampling algorithm for sampling T independent elements from any discrete probability distribution $(p_1, \ldots, p_n)$ in time $O(\sqrt{Tn} \cdot \mathrm{EO})$, instead of $O(T\sqrt{n} \cdot \mathrm{EO})$. This algorithm is described in the next paragraph. It leads to our second main result.

Statement of Theorem 5 *There is a quantum algorithm that, given a submodular function $F : 2^V \to [-1,1]$ and $\epsilon > 0$, computes a set $\bar{S}$ such that $\mathbb{E}[F(\bar{S})] \leq \min_{S \subseteq V} F(S) + \epsilon$ in time $\widetilde{O}(n^{5/4}/\epsilon^{5/2} \cdot \log(\frac{1}{\epsilon}) \cdot \mathrm{EO})$.*

Quantum multi-sampling algorithm. We now sketch the algorithm for sampling T elements from $p = (p_1, \ldots, p_n)$ in time $O(\sqrt{Tn} \cdot \mathrm{EO})$ (here EO is the time complexity of a quantum evaluation oracle $\mathcal{O}_p$ satisfying $\mathcal{O}_p(|i\rangle|0\rangle) = |i\rangle|p_i\rangle$ for all i). First, we find the set S of all the coordinates $i \in [n]$ where p_i is larger than $1/T$. Since there are at most T values to find, S can be computed in time $O(\sqrt{Tn} \cdot \mathrm{EO})$ using Grover search. Then, we load in time $O(T \cdot \mathrm{EO})$ the conditional distribution $(p_i/p_S)_{i \in S}$, where $p_S = \sum_{i \in S} p_i$, into a classical data structure [35] that supports fast sampling from $(p_i/p_S)_{i \in S}$ in time $O(1)$.

On the other hand, we can sample from the complement distribution $(p_i/(1-p_S))_{i \notin S}$ using quantum state preparation of $\frac{1}{\sqrt{1-p_S}} \sum_{i \notin S} \sqrt{p_i}|i\rangle$ in time $O(\sqrt{n \cdot \max_{i \notin S} p_i/(1-p_S)} \cdot \mathrm{EO}) = O(\sqrt{n/(T(1-p_S))} \cdot \mathrm{EO})$. Now, each of the T samples is obtained by first flipping a coin that lands head with probability p_S, and then sampling $i \in S$ from the classical data structure (head case) or $i \notin S$ by quantum state preparation (tail case). The total expected time is $O\big(Tp_S \cdot 1 + T(1-p_S) \cdot \sqrt{n/(T(1-p_S))} \cdot \mathrm{EO}\big) = O(\sqrt{Tn} \cdot \mathrm{EO})$ (assuming $T < n$). Additional technicalities, arising from the fact that $(p_1, \ldots, p_n) = (u_1/\|u\|_1, \ldots, u_n/\|u\|_1)$ may be given as an unormalized vector $(u_1, \ldots, u_n)$, are also discussed in this paper (Section 3).

Statement of Theorem 2 *There is a quantum algorithm that, given an integer $1 < T < n$, a real $0 < \delta < 1$, and an evaluation oracle to a discrete probability distribution $\mathcal{D} = (p_1, \ldots, p_n)$, outputs T independent samples from $\mathcal{D}$ in expected time $O\big(\sqrt{Tn} \log(1/\delta) \cdot \mathrm{EO}\big)$ with probability $1-\delta$.*

1.5 Organization of the Paper

Our algorithms are presented in a modular way. In Section 4, we give the common framework to the classical and quantum algorithms. Then, we specialize it to each setting in Sections 5.2 and 5.3 respectively. A data structure, which is common to both models, is described in Section 5.1. The robustness of the stochastic subgradient descent (Proposition 3), and the quantum algorithm for sampling from discrete probability distributions (Section 3) can be read independently from the rest of the paper. The reader interested only in the classical algorithm can skip Sections 3 and 5.3.

1.6 Recent Improvement

After having finished this paper, we have been informed through personal communication that Axelrod, Liu and Sidford have discovered a classical nearly linear time algorithm for approximate submodular function minimization [36]. Their result, like ours, improves on the work of Chakrabarty et al. [2], and it outperforms both our algorithms.

1.7 Open Questions

It seems to us that, in order to achieve a quantum speed-up over the best classical algorithms for approximate [36] or exact [1] submodular function minimization, one would most likely have to speed-up the gradient descent or cutting plane methods respectively. This latter problem is notoriously open in the quantum setting. Another more amenable question is whether the $\Omega(n)$ lower bound for exact minimization [37] carries over to the quantum oracle model, and whether $\Omega(n/\epsilon^2)$ is a lower bound in the approximate case. Finally, what can be other applications of our quantum multi-sampling algorithms beyond submodular function minimization?

2 Preliminaries

Notations. Let $[n] = \{1, \ldots, n\}$. Given a vector $u \in \mathbb{R}^n$ and a positive integer p, we let $\|u\|_p = \big(\sum_{i \in [n]} |u_i|^p\big)^{1/p}$ be the ℓ_p-norm of u, and $\|u\|_\infty = \max_{i \in [n]} |u_i|$ be the largest entry (in absolute value). We say that u is k-sparse if it has at most k non-zero entries. We denote by $u_+ \in \mathbb{R}^n$ (resp. $u_- \in \mathbb{R}^n$) the vector obtained from u by replacing its negative (resp. positive) entries with 0 (thus, $u = u_+ + u_-$). Given two vectors $u, u' \in \mathbb{R}^n$, we use $u \geq u'$

(resp. $u \leq u'$) to denote that $u - u' \in \mathbb{R}^n_+$ (resp. $u - u' \in \mathbb{R}^n_-$). We also let $\operatorname{sgn}(u)$ to be 1 if $u \geq 0$, and -1 otherwise. Given a set $S \subseteq [n]$, we denote by $u_S \in \mathbb{R}^{|S|}$ the subvector $(u_i)_{i \in S}$ of u made of the values at coordinates $i \in S$. If u is a non-zero vector, we define $\mathcal{D}_u$ to be the probability distribution $\left(\frac{|u_1|}{\|u\|_1}, \dots, \frac{|u_n|}{\|u\|_1}\right)$ on $[n]$. Finally, we let $\vec{1}_i \in \mathbb{R}^n$ be the indicator vector with a 1 at position $i \in [n]$ and 0 elsewhere.

Lovász extension. A submodular function F is a set function $F : 2^V \to \mathbb{R}$, over some ground set V of size n, that satisfies the diminishing returns property: for every $A \subseteq B \subseteq V$ and for every $i \notin B$, the inequality $F(A \cup \{i\}) - F(A) \geq F(B \cup \{i\}) - F(B)$ holds. For convenience, and without loss of generality, we assume that $V = [n]$ and $F(\varnothing) = 0$ (this can be enforced by observing that $S \mapsto F(S) - F(\varnothing)$ is still a submodular function). The Lovász extension $f : [0,1]^n \to \mathbb{R}$ is a convex relaxation of F to the hypercube $[0,1]^n$. Before describing it, we present a canonical way to associate a permutation P with each $x \in [0,1]^n$.

Definition 1 *Given a permutation $P = (P_1, \dots, P_n)$ of $[n]$, we say that P is* consistent *with $x \in \mathbb{R}^n$ if $x_{P_1} \geq x_{P_2} \geq \dots \geq x_{P_n}$, and $P_{i+1} > P_i$ when $x_{P_i} = x_{P_{i+1}}$ for all i. We also denote $P[i] = \{P_1, \dots, P_i\} \subseteq [n]$ the set of the first i elements of P, and $P[0] = \varnothing$.*

As an example, the permutation P consistent with $x = (0.3, 0.2, 0.3, 0.1)$ is $P = (1, 3, 2, 4)$.

Definition 2 *Given a submodular function $F : 2^V \to \mathbb{R}$ over $V = [n]$, the* Lovász extension *$f : [0,1]^n \to \mathbb{R}$ of F is defined for all $x \in [0,1]^n$ by $f(x) = \sum_{i \in [n]} (F(P[i]) - F(P[i-1])) \cdot x_{P_i}$ where P is the permutation consistent with x. The* Lovász subgradient *$g(x) \in \mathbb{R}^n$ at $x \in [0,1]^n$ is defined by $g(x)_{P_i} = F(P[i]) - F(P[i-1])$ for all $i \in [n]$.*

The following standard properties of the Lovász extension [12, 3, 38] will be used in this paper.

Proposition 1 *The Lovász extension f of a submodular function F is a convex function. Moreover, given $x \in [0,1]^n$ and the permutation P consistent with x, we have*

1. **(Subgradient)** *For all $y \in [0,1]^n$, $\langle g(x), x - y \rangle \geq f(x) - f(y)$.*
2. **(Minimizers)** $\min_{i \in [n]} F(P[i]) \leq f(x)$ *and* $\min_{S \subseteq V} F(S) = \min_{y \in [0,1]^n} f(y)$.
3. **(Boundedness)** *If the range of F is $[-1, 1]$ then $\|g(x)\|_2 \leq \|g(x)\|_1 \leq 3$.*

Observe that the second property gives an explicit way to convert any $\bar{x} \in [0,1]^n$ such that $f(\bar{x}) \leq \min_{x \in [0,1]^n} f(x) + \epsilon$ into a set $\bar{S} \subseteq V$ such that $F(\bar{S}) \leq \min_{S \subseteq V} F(S) + \epsilon$. Consequently, we can focus on ϵ-additive minimization of the Lovász extension in the rest of the paper.

Models of Computation. We describe the two models of computation used in this paper. Although the Lovász extension is a continuous function, given $x \in [0,1]^n$ it is sufficient to evaluate F on the sets $P[1], \dots, P[n]$ to compute $f(x)$, where P is the permutation consistent with x. The same holds for the Lovász subgradient. Consequently, given P, it is natural to define an evaluation oracle that given i returns $F(P[i])$. The input i to this oracle is encoded over $O(\log n)$ bits, whereas representing each of the sets $P[i]$ as an indicator vector over $\{0,1\}^n$ would require n bits.

- *Classical Model.* We use the same model as described in [2]. The submodular function F can be accessed via an evaluation oracle that takes as input an integer $i \in [n]$ and a linked list storing a permutation P of $[n]$, and returns the value of $F(P[i])$. We denote by EO the cost of one evaluation query to the oracle.

- *Quantum Model.* We extend the above model to the quantum setting in a standard way. Given a permutation P of $[n]$ stored in a linked list, we assume that we have access to a unitary operator $\mathcal{O}_P$ that, given $i \in [n]$, satisfies $\mathcal{O}_P(|i\rangle|0\rangle) = |i\rangle|F(P[i])\rangle$, where the second register holds a binary representation of $F(P[i])$ with some finite precision. We denote by EO the cost of one evaluation query to $\mathcal{O}_P$.

The Lovász extension $f(x)$ at x can be evaluated in time $O(n \log n + n \cdot \mathrm{EO})$ in the above models.

3 Quantum Multi-Sampling for Discrete Probability Distributions

We study the problem of generating T independent samples from a discrete probability distribution $\mathcal{D}_u = \left(\frac{|u_1|}{\|u\|_1}, \ldots, \frac{|u_n|}{\|u\|_1}\right)$ on $[n]$, where $u = (u_1, \ldots, u_n) \in \mathbb{R}^n$ is a non-zero vector given as an evaluation oracle. This task is a fundamental part of Monte Carlo methods and discrete events simulation [39, 40]. Here, it will be used to construct randomized estimators of the Lovász subgradient in Sections 5.2 and 5.3. In this section, EO denotes the time complexity of an evaluation oracle to u. In the classical setting, this oracle must return u_i given $i \in [n]$, whereas in the quantum setting it is a unitary operator $\mathcal{O}_u$ satisfying $\mathcal{O}_u(|i\rangle|0\rangle) = |i\rangle|u_i\rangle$ for all i.

The above problem has been thoroughly investigated in the classical setting [39, 40], where it can be solved in time $O(n \cdot \mathrm{EO} + T)$ using the alias method. We present this result below, as it will be part of our quantum algorithm later.

Lemma 1 ([41, 35]) *There is a classical algorithm that, given an evaluation oracle to a non-zero vector $u \in \mathbb{R}^n$, constructs in time $O(n \cdot \mathrm{EO})$ a data structure from which one can output as many independent samples from $\mathcal{D}_u$ as desired, each in time $O(1)$.*

In the quantum setting, it is a well-known result that one sample from $\mathcal{D}_u$ can be obtained by preparing the state $\sum_{i \in [n]} \sqrt{\frac{|u_i|}{\|u\|_1}} |i\rangle$ with Amplitude Amplification and measuring the $|i\rangle$ register.

Lemma 2 ([42]) *There is a quantum algorithm that, given an evaluation oracle to a non-zero vector $u \in \mathbb{R}^n$ and a value $M \geq \|u\|_\infty$, outputs one sample from $\mathcal{D}_u$ in expected time $O\left(\sqrt{\frac{nM}{\|u\|_1}} \cdot \mathrm{EO}\right)$.*

Note that the maximum $M = \|u\|_\infty$ of any vector $u \in \mathbb{R}^n$ can be computed with high probability using Dürr-Høyer's algorithm [43] in time $O(\sqrt{n} \cdot \mathrm{EO})$, in which case we have $\sqrt{nM/\|u\|_1} \leq \sqrt{n}$. Then, by simply repeating the above algorithm T times, one can obtain T samples in time $O(T\sqrt{n} \cdot \mathrm{EO})$. Our main contribution (Algorithm 1) is to improve this time complexity to $O(\sqrt{Tn} \cdot \mathrm{EO})$. If the normalization factor $\|u\|_1$ is unknown, we will only be able to sample from a distribution $\mathcal{D}_u(\Gamma, S)$ close to $\mathcal{D}_u$ that is defined below. Here, $\Gamma > 0$ acts as a placeholder for an estimate of $\|u\|_1$ and $S \subseteq [n]$ is meant to contain the indices i where $|u_i|$ is larger than Γ/T.

Definition 3 *Consider a non-zero vector $u \in \mathbb{R}^n$. Fix a real number $\Gamma > 0$ and a set $S \subseteq [n]$ such that $\Gamma \geq \|u_S\|_1$. We define $\mathcal{D}_u(\Gamma, S)$ to be the distribution that outputs $i \in [n]$ with probability*

$$\begin{cases} \frac{|u_i|}{\Gamma} & \text{if } i \in S \\ \frac{|u_i|}{\Gamma} + \left(1 - \frac{\|u\|_1}{\Gamma}\right) \frac{|u_i|}{\|u_{[n]\setminus S}\|_1} = \left(1 - \frac{\|u_S\|_1}{\Gamma}\right) \frac{|u_i|}{\|u_{[n]\setminus S}\|_1} & \text{if } i \in [n] \setminus S. \end{cases}$$

Note that if $\Gamma = \|u\|_1$ then $\mathcal{D}_u(\|u\|_1, S) = \mathcal{D}_u$, which is independent of S.

We now prove that Algorithm 1 runs in time $O(\sqrt{Tn} \cdot \mathrm{EO})$ when Γ is sufficiently close to $\|u\|_1$ and $S = \{i \in [n] : |u_i| \geq \Gamma/T\}$. We will explain later how to find such parameters in time $O(\sqrt{Tn} \cdot \mathrm{EO})$.

Algorithm 1 Sampling T elements from $\mathcal{D}_u(\Gamma, S)$.

Input: a non-zero vector $u \in \mathbb{R}^n$, an integer $1 < T < n$, a real $\Gamma > 0$ and a set $S \subseteq [n]$ such that $\Gamma \geq \|u_S\|_1$, the value $M = \|u_{[n]\setminus S}\|_\infty$.
Output: a sequence $(i_1, \ldots, i_T) \in [n]^T$.

```
1: Construct the data structure associated with $u_S = (u_i)_{i \in S}$ in Lemma 1, and compute $\|u_S\|_1$.
2: for $t = 1, \ldots, T$ do
3:     Sample $b_t \in \{0, 1\}$ from the Bernoulli distribution of parameter $p = \frac{\|u_S\|_1}{\Gamma}$.
4:     If $b_t = 1$, sample $i_t \sim \mathcal{D}_{u_S}$ using the data structure built at step 1.
5:     If $b_t = 0$, sample $i_t \sim \mathcal{D}_{u_{[n]\setminus S}}$ using Lemma 2 with input $u_{[n]\setminus S}$ and $M$.
6: Output $(i_1, \ldots, i_T)$.
```

Theorem 1 *The output $(i_1, \ldots, i_T) \in [n]^T$ of Algorithm 1 consists of T independent samples from the distribution $\mathcal{D}_u(\Gamma, S)$. Moreover, if $|\Gamma - \|u\|_1| \leq \|u\|_1/\sqrt{T}$ and $S = \{i \in [n] : |u_i| \geq \Gamma/T\}$ then the expected run-time of the algorithm is $O(\sqrt{Tn} \cdot \mathrm{EO})$.*

Proof. At each execution of lines 2-5, the probability to sample $i \in S$ is $\frac{\|u_S\|_1}{\Gamma} \cdot \frac{|u_i|}{\|u_S\|_1} = \frac{|u_i|}{\Gamma}$ and the probability to sample $i \in [n] \setminus S$ is $\left(1 - \frac{\|u_S\|_1}{\Gamma}\right) \frac{|u_i|}{\|u_{[n]\setminus S}\|_1}$. This is the distribution $\mathcal{D}_u(\Gamma, S)$.

We now analyze the time complexity. Line 1 takes time $O(|S| \cdot \mathrm{EO})$. Each execution of line 4 takes time $O(1)$, and each execution of line 5 takes time $O\left(\sqrt{n \cdot \|u_{[n]\setminus S}\|_\infty / \|u_{[n]\setminus S}\|_1} \cdot \mathrm{EO}\right)$ (according to Lemma 2). Thus, the expected run-time of the algorithm is

$$O\left(|S| \cdot \mathrm{EO} + T \frac{\|u_S\|_1}{\Gamma} \cdot 1 + T \left(1 - \frac{\|u_S\|_1}{\Gamma}\right) \cdot \sqrt{\frac{n \cdot \|u_{[n]\setminus S}\|_\infty}{\|u_{[n]\setminus S}\|_1}} \cdot \mathrm{EO}\right).$$

Assume that $|\Gamma - \|u\|_1| \leq \|u\|_1/\sqrt{T}$ and $S = \{i \in [n] : |u_i| \geq \Gamma/T\}$. Since $T \geq 2$, it follows that $\Gamma \geq \|u\|_1/4$ and $|S| \leq 4T$. Consequently, $1 - \frac{\|u_S\|_1}{\Gamma} \leq \left(1 + \frac{1}{\sqrt{T}}\right) \frac{\|u\|_1}{\Gamma} - \frac{\|u_S\|_1}{\Gamma} \leq \frac{\|u_{[n]\setminus S}\|_1}{\Gamma} + \frac{4}{\sqrt{T}}$. Moreover, $\|u_{[n]\setminus S}\|_\infty \leq \min(\Gamma/T, \|u_{[n]\setminus S}\|_1)$. Thus, the expected run-time is

$$O\left(T \cdot \mathrm{EO} + T \left(\frac{\|u_{[n]\setminus S}\|_1}{\Gamma} + \frac{1}{\sqrt{T}}\right) \cdot \sqrt{\frac{n \cdot \min(\Gamma/T, \|u_{[n]\setminus S}\|_1)}{\|u_{[n]\setminus S}\|_1}} \cdot \mathrm{EO}\right) = O\left(\sqrt{Tn} \cdot \mathrm{EO}\right).$$

□

The above result is optimal, as can be shown by a simple reduction from the T-search problem. We now explain how to find the values Γ, S and $\|u_{[n]\setminus S}\|_\infty$ needed by Algorithm 1. First, if $\|u\|_1$ is known, we can assume without loss of generality that $\|u\|_1 = 1$. In this case, we obtain T samples from $\mathcal{D}_u = (p_1, \ldots, p_n)$ as follows.

Theorem 2 *There is a quantum algorithm that, given an integer* $1 < T < n$*, a real* $0 < \delta < 1$*, and an evaluation oracle to a discrete probability distribution* $\mathcal{D} = (p_1, \ldots, p_n)$*, outputs* T *independent samples from* $\mathcal{D}$ *in expected time* $O\big(\sqrt{Tn}\log(1/\delta)\cdot \mathrm{EO}\big)$ *with probability* $1 - \delta$*.*

Proof. The set $S = \{i \in [n] : |p_i| \geq 1/T\}$ and the value $M = \|p_{[n]\setminus S}\|_\infty$ can be computed with probability $1-\delta$ using Grover search and Dürr-Høyer's algorithm [43] in time $O(\sqrt{Tn}\log(1/\delta)\cdot \mathrm{EO})$ and $O(\sqrt{n}\log(1/\delta)\cdot \mathrm{EO})$ respectively. Then, conditioned on these two values to be correct, Algorithm 1 outputs T independent samples from $\mathcal{D}$ in expected time $O(\sqrt{Tn}\cdot \mathrm{EO})$ (where we use $\Gamma = 1$). □

If $\|u\|_1$ is unknown (as it will be the case in our applications), we will need the next result about Amplitude Estimation [44] to approximate its value.

Lemma 3 *There is a quantum algorithm that, given an evaluation oracle to a non-zero vector* $u \in \mathbb{R}^n$*, a value* $M \geq \|u\|_\infty$ *and two reals* $0 < \epsilon, \delta < 1$*, outputs a real* Γ *such that* $|\Gamma - \|u\|_1| \leq \epsilon\|u\|_1$ *with probability* $1-\delta$*. The expected run-time of this algorithm is* $O\Big(\frac{1}{\epsilon}\sqrt{\frac{nM}{\|u\|_1}}\log(1/\delta)\cdot \mathrm{EO}\Big)$*.*

Proof. Define $V_{u,M}$ to be a unitary operator such that

$$V_{u,M}(|0\rangle|0\rangle) = \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle\left(\sqrt{\frac{|u_i|}{M}}|0\rangle + \sqrt{1-\frac{|u_i|}{M}}|1\rangle\right) = \sqrt{\frac{\|u\|_1}{nM}}|\psi_u\rangle|0\rangle + \sqrt{1-\frac{\|u\|_1}{nM}}|\phi_u\rangle|1\rangle$$

where $|\psi_u\rangle = \sum_{i\in[n]}\sqrt{\frac{|u_i|}{\|u\|_1}}|i\rangle$, and $|\phi_u\rangle$ is some unit vector. $V_{u,M}$ can be constructed with two quantum queries to u and a controlled rotation (see also [45] for an alternative construction). Now, using the Amplitude Estimation algorithm [44, Theorem 12] on $V_{u,M}$ with accuracy ϵ, we get an estimate γ such that $|\gamma - \|u\|_1/(nM)| \leq \epsilon\|u\|_1/(nM)$ with probability $2/3$ in expected time $O\Big(\frac{1}{\epsilon}\sqrt{\frac{nM}{\|u\|_1}}\cdot \mathrm{EO}\Big)$. The success probability can be increased to $1-\delta$ by a standard Chernoff bound argument at an extra cost factor $\log(1/\delta)$. Finally, we take $\Gamma = nM\gamma$. □

The construction of the setup parameters (Γ, S, M) is described in Algorithm 2. We need to be careful that $\Gamma \geq \|u_S\|_1$, otherwise $\mathcal{D}_u(\Gamma, S)$ is not a probability distribution. The parameter ϵ controls the closeness of $\mathcal{D}_u(\Gamma, S)$ to $\mathcal{D}_u$. We have $\epsilon' = \min(1/\sqrt{T}, \epsilon)$ to guarantee that $|\Gamma - \|u\|_1| \leq (1/\sqrt{T})\|u\|_1$. The setup cost is dominated by $O(\sqrt{n}/\epsilon)$ if $\epsilon \leq 1/\sqrt{T}$.

Proposition 2 *The output* (Γ, S, M) *of Algorithm 2 satisfies* $\Gamma \geq \|u_S\|_1$*,* $|\Gamma - \|u\|_1| \leq \min(1/\sqrt{T}, \epsilon)\|u\|_1$*,* $S = \{i \in [n] : |u_i| \geq \Gamma/T\}$ *and* $M = \|u_{[n]\setminus S}\|_\infty$ *with probability* $1-\delta$*. The expected run-time of this algorithm is* $O\big((\sqrt{Tn} + \sqrt{n}/\epsilon)\log(1/\delta)\cdot \mathrm{EO}\big)$*.*

Proof. We first assume that all steps of the algorithm succeed and do not abort. In this case, we have $|\hat{\Gamma} - \|u\|_1| \leq \epsilon'\|u\|_1$. Thus, $\Gamma = \max\{\|u_{\hat{S}}\|_1, \hat{\Gamma}\} \geq \max\{\|u_S\|_1, (1-\epsilon')\|u\|_1\}$ and $\Gamma \leq (1+\epsilon')\|u\|_1$. Moreover, $S = \{i \in [n] : |u_i| \geq \Gamma/T\}$ since $\{i \in [n] : |u_i| \geq \Gamma/T\} \subseteq \{i \in [n] : |u_i| \geq \hat{\Gamma}/T\} = \hat{S}$.

We now study the time needed by lines 1-5 to succeed with probability $1-\delta$. If we omit the $\log(1/\delta)\cdot \mathrm{EO}$ factors, then there exist four absolute constants c_1, c_2, c_3 and c_4 such that lines 1 and 5 need time $c_1 \cdot \sqrt{n}$, line 2 needs time $c_2 \cdot \frac{1}{\epsilon'}\sqrt{nL/\|u\|_1} \leq c_2 \cdot (\sqrt{Tn} + \sqrt{n}/\epsilon)$

Algorithm 2 Construction of the setup parameters (Γ, S, M).

Input: a non-zero vector $u \in \mathbb{R}^n$, an integer $1 < T < n$, two reals $0 < \epsilon, \delta < 1$.
Output: a real Γ, a set $S \subseteq [n]$, a value M.

The subroutines below are run with failure parameter $\delta/4$. The algorithm aborts and outputs *fail* if any step takes time greater than $c \cdot (\sqrt{Tn} + \sqrt{n}/\epsilon)\log(1/\delta)$ (where c is a constant to be specified in the proof of Proposition 2).

```
1: Run Dürr-Høyer's algorithm [43] to compute $\|u\|_\infty$. Denote the result by $L$.
2: Compute an estimate $\hat{\Gamma}$ of $\|u\|_1$ with relative error $\epsilon' = \min(1/\sqrt{T}, \epsilon)$ using $L$ and Lemma 3.
3: Run the Grover search algorithm [46] on $u$ to find all the indices $i$ such that $|u_i| \geq \hat{\Gamma}/T$. Denote the result by $\hat{S} \subseteq [n]$.
4: Compute $\|u_{\hat{S}}\|_1$ and set $\Gamma = \max\{\|u_{\hat{S}}\|_1, \hat{\Gamma}\}$. Compute $S = \{i \in \hat{S} : |u_i| \geq \Gamma/T\}$.
5: Run Dürr-Høyer's algorithm [43] to compute $\|u_{[n]\setminus S}\|_\infty$. Denote the result by $M$.
6: Output $(\Gamma, S, M)$.
```

(according to Lemma 3, and since $L = \|u\|_\infty \leq \|u\|_1$ if line 1 succeeds), line 3 needs time $c_3 \cdot \sqrt{Tn}$ (since $\hat{S} \leq 4T$ if $\hat{S} = \{i \in [n] : |u_i| \geq \hat{\Gamma}/T\}$ and $\hat{\Gamma} \geq (1 - \epsilon')\|u\|_1 \geq \|u\|_1/4)$ and line 4 needs time $c_4|\hat{S}| \leq 4c_4T$. Consequently, if we take $c = \max\{c_1, c_2, c_3, 4c_4\}$, the algorithm does not abort and succeeds with probability $1 - \delta$. □

4 Framework for Approximate Submodular Minimization

In this section, we construct our new low-variance estimate of the Lovász subgradient, and we apply the stochastic subgradient descent algorithm on it to minimize the Lovász extension. The stochastic subgradient descent method is a general algorithm for approximating the minimum value of a convex function f that is not necessarily differentiable (as it is the case for the Lovász extension). It uses the concept of *subgradients* (or *subderivatives*) of f, which is defined as follows.

Definition 4 *Given a convex function $f : C \to \mathbb{R}$ over $C \subset \mathbb{R}^n$ and a point $x \in C$, we say that $g \in \mathbb{R}^n$ is a* subgradient *of f at x if $\langle g, x - y\rangle \geq f(x) - f(y)$ for all $y \in C$. The set of all subgradients at x is denoted by $\partial f(x)$.*

Normally, the stochastic subgradient descent method requires to compute a sequence $(\widetilde{g}^{(t)})_t$ of *unbiased* subgradient estimates at certain points $(x^{(t)})_t$, which means that $\mathbb{E}[\widetilde{g}^{(t)} \mid x^{(t)}] \in \partial f(x^{(t)})$. In the next proposition, we generalize this method to ϵ-noisy estimates satisfying only $\|\mathbb{E}[\widetilde{g}^{(t)} \mid x^{(t)}] - g^{(t)}\|_1 \leq \epsilon$ for some $g^{(t)} \in \partial f(x^{(t)})$. In the case $\epsilon = 0$, our analysis recovers the standard error bound [47].

Proposition 3 (Noisy Stochastic Subgradient Descent) *Let $f : C \to \mathbb{R}$ be a convex function over a compact convex set $C \subset \mathbb{R}^n$, and $\eta > 0$. Consider two sequences of random variables $(x^{(t)})_t$ and $(\widetilde{g}^{(t)})_t$ such that $x^{(0)} = \operatorname{argmin}_{x \in C}\|x\|_2$, $x^{(t+1)} = \operatorname{argmin}_{x \in C}\|x - (x^{(t)} - \eta\widetilde{g}^{(t)})\|_2$, and*

$$\left\|\mathbb{E}\left[\widetilde{g}^{(t)} \mid x^{(t)}\right] - g^{(t)}\right\|_1 \leq \epsilon \text{ for some } g^{(t)} \in \partial f(x^{(t)}),$$

for all $t \geq 0$. Fix $x^\star \in \operatorname{argmin}_{x \in C} f(x)$ and let $L_2, L_\infty, B \in \mathbb{R}$ be such that $\|x - x^\star\|_2 \leq L_2$, $\|x - x^\star\|_\infty \leq L_\infty$ and $\mathbb{E}\left[\|\widetilde{g}^{(t)}\|_2^2\right] \leq B^2$, for all $x \in C$ and $t \geq 0$. Then, for any integer N, the average point $\bar{x} = \frac{1}{N}\sum_{t=0}^{N-1} x^{(t)}$ satisfies $\mathbb{E}[f(\bar{x})] \leq f(x^\star) + \frac{L_2^2}{2\eta N} + \frac{\eta}{2}B^2 + \epsilon L_\infty$.

Proof. Let $(g^{(t)})_t$ be such that $g^{(t)} \in \partial f(x^{(t)})$ and $\|\mathbb{E}[\widetilde{g}^{(t)} \mid x^{(t)}] - g^{(t)}\|_1 \leq \epsilon$. Then,

$$
\begin{aligned}
\|x^{(t+1)} - x^\star\|_2^2 &= \left\| \operatorname*{argmin}_{x \in C} \|x - (x^{(t)} - \eta \widetilde{g}^{(t)})\|_2 - x^\star \right\|_2^2 \\
&\leq \|x^{(t)} - \eta \widetilde{g}^{(t)} - x^\star\|_2^2 \quad \text{by property of the projection onto } C \\
&= \|x^{(t)} - x^\star\|_2^2 - 2\eta \langle \widetilde{g}^{(t)}, x^{(t)} - x^\star \rangle + \eta^2 \|\widetilde{g}^{(t)}\|_2^2 \\
&= \|x^{(t)} - x^\star\|_2^2 - 2\eta \langle g^{(t)}, x^{(t)} - x^\star \rangle - 2\eta \langle \widetilde{g}^{(t)} - g^{(t)}, x^{(t)} - x^\star \rangle + \eta^2 \|\widetilde{g}^{(t)}\|_2^2 \\
&\leq \|x^{(t)} - x^\star\|_2^2 - 2\eta (f(x^{(t)}) - f(x^\star)) - 2\eta \langle \widetilde{g}^{(t)} - g^{(t)}, x^{(t)} - x^\star \rangle + \eta^2 \|\widetilde{g}^{(t)}\|_2^2
\end{aligned}
$$

where the last line is by the definition of a subgradient. We now take the expectation of the above formula. Using the law of total expectation, we have $\mathbb{E}\big[\langle \widetilde{g}^{(t)} - g^{(t)}, x^{(t)} - x^\star \rangle\big] = \mathbb{E}\big[\langle \mathbb{E}\big[\widetilde{g}^{(t)} \bigm| x^{(t)}\big] - g^{(t)}, x^{(t)} - x^\star \rangle\big]$ and by Hölder's inequality $\big|\langle \mathbb{E}\big[\widetilde{g}^{(t)} \bigm| x^{(t)}\big] - g^{(t)}, x^{(t)} - x^\star \rangle\big| \leq \|\mathbb{E}\big[\widetilde{g}^{(t)} \bigm| x^{(t)}\big] - g^{(t)}\|_1 \cdot \|x^{(t)} - x^\star\|_\infty \leq \epsilon L_\infty$. Consequently,

$$
\mathbb{E}\Big[\|x^{(t+1)} - x^\star\|_2^2\Big] - \mathbb{E}\Big[\|x^{(t)} - x^\star\|_2^2\Big] \leq -2\eta \mathbb{E}\Big[f(x^{(t)}) - f(x^\star)\Big] + 2\eta\epsilon L_\infty + \eta^2 B^2
$$

from which we obtain a bound for $\mathbb{E}[f(x^{(t)})]$. Finally, we upper bound the expected value of the function at the average point $\bar{x}$ as

$$
\begin{aligned}
\mathbb{E}[f(\bar{x})] &\leq \frac{1}{N} \sum_{t=0}^{N-1} \mathbb{E}\Big[f(x^{(t)})\Big] \quad \text{by convexity} \\
&\leq f(x^\star) + \frac{1}{N} \sum_{t=0}^{N-1} \frac{1}{2\eta} \left(\mathbb{E}\Big[\|x^{(t)} - x^\star\|_2^2\Big] - \mathbb{E}\Big[\|x^{(t+1)} - x^\star\|_2^2\Big] \right) + \frac{\eta}{2} B^2 + \epsilon L_\infty \\
&= f(x^\star) + \frac{1}{2\eta N} \left(\mathbb{E}\Big[\|x^{(0)} - x^\star\|_2^2\Big] - \mathbb{E}\Big[\|x^{(N)} - x^\star\|_2^2\Big] \right) + \frac{\eta}{2} B^2 + \epsilon L_\infty \\
&\leq f(x^\star) + \frac{L_2^2}{2\eta N} + \frac{\eta}{2} B^2 + \epsilon L_\infty
\end{aligned}
$$

where we have used the telescoping property of the sum in the third line. □

In the rest of the paper, f denotes the Lovász extension and g denotes the Lovász subgradient. Our main result of this section (Algorithm 3) consists in constructing the sequence of noisy subgradient estimates needed in the above proposition. To trade off the cost of computing the subgradient exactly and decreasing the variance, we rely on two procedures that provide different guarantees on the estimates they return. In this section, we do not explain how to implement these two procedures. Instead, we describe in Assumptions 1 and 2 the main properties they must satisfy.

Our first assumption is the existence of a procedure GSample that can produce a batch of T estimates of the Lovász subgradient $g(x)$ at any point $x \in [0,1]^n$. This is intended to be a simple but expensive procedure, which can be used only sparingly. Indeed, it will need time $O((n+T) \cdot \mathrm{EO})$ or $O((\sqrt{nT} + \sqrt{n}/\epsilon) \cdot \mathrm{EO})$ to be implemented in the classical or quantum settings respectively (Propositions 5 and 8).

Assumption 1 (Gradient Sampling) *There is a procedure* GSample(x, T, ϵ) *that, given* $x \in [0,1]^n$*, an integer* T *and a real* $\epsilon > 0$*, outputs* T *vectors* $\widetilde{g}^1, \ldots, \widetilde{g}^T$ *such that, for all* j*, (1)* $\widetilde{g}^j$ *is* 1*-sparse, (2)* $\big\|\mathbb{E}[\widetilde{g}^j \mid \widetilde{g}^1, \ldots, \widetilde{g}^{j-1}, x] - g(x)\big\|_1 \leq \epsilon$ *and (3)* $\|\widetilde{g}^j\|_2 \leq 4$*. Moreover, the time complexity of this procedure is a function* $c_{\mathsf{GS}}(T, \epsilon)$ *of* T *and* ϵ*.*

Our second assumption is the existence of a more subtle procedure GDSample that can estimate the difference $g(y) - g(x)$ between the Lovász subgradients at two points x and y. This procedure will rely on intrinsic properties of submodular functions and require maintaining a particular data structure (Section 5.1). On the other hand, when the difference $e = y - x$ is k-sparse, it will need time only $\widetilde{O}(k \cdot \mathrm{EO})$ or $\widetilde{O}(\sqrt{k}/\epsilon \cdot \mathrm{EO})$ to be implemented in the classical or quantum settings respectively (Propositions 6 and 9).

Assumption 2 (Gradient Difference Sampling) *There is a procedure* $\mathsf{GDSample}(x, e, \epsilon)$ *that, given* $x \in [0,1]^n$*, a* k*-sparse vector* e *such that* $x+e \in [0,1]^n$ *and* $e \geq 0$ *or* $e \leq 0$*, and a real* $\epsilon > 0$*, outputs a vector* $\widetilde{d}$ *such that, (1)* $\widetilde{d}$ *is* 1*-sparse, (2)* $\|\mathbb{E}[\widetilde{d} \mid x, e] - (g(x+e) - g(x))\|_1 \leq \epsilon$ *and (3)* $\|\widetilde{d}\|_2 \leq 7$*. Moreover, the time complexity of this procedure is a function* $c_{\mathsf{GDS}}(k, \epsilon)$ *of* k *and* ϵ*.*

We combine the two procedures to construct the sequence $(\widetilde{g}^{(t)})_t$ of subgradient estimates (Algorithm 3). The construction depends on a "loop parameter" T that balances the cost between using $\mathsf{GSample}$ and $\mathsf{GDSample}$. Every T steps, when $t = 0 \bmod T$, the procedure $\mathsf{GSample}(x^{(t)}, T, \epsilon)$ returns T estimates $\widetilde{g}^{(t,0)}, \ldots, \widetilde{g}^{(t,T-1)}$ of the Lovász subgradient at the current point $x^{(t)}$. Each value $\widetilde{g}^{(t,\tau)}$ is combined at time $t + \tau$, where $0 \leq \tau \leq T-1$, with an estimate $\widetilde{d}^{(t+\tau)}$ of the subgradient difference $g(x^{(t+\tau)}) - g(x^{(t)})$. The sum $\widetilde{g}^{(t+\tau)} = \widetilde{g}^{(t,\tau)} + \widetilde{d}^{(t+\tau)}$ is our estimate of $g(x^{(t+\tau)})$. The sparsity of $x^{(t+\tau)} - x^{(t)}$ will increase linearly in τ, which justifies reusing $\mathsf{GSample}$ every T steps to restore it to a small value. Notice that, according to Assumption 2, the procedure $\mathsf{GDSample}$ can estimate the subgradient difference $d = g(y) - g(x)$ only if $e = y - x$ is either non-negative or non-positive. Thus, in step 2.(c) of the algorithm, we split $e = e_+ + e_-$ into its positive and negative entries and we estimate $d_+ = g(x + e_+) - g(x)$ and $d_- = g(x + e_+ + e_-) - g(x + e_+)$ separately. In the next theorem, we show that $(\widetilde{g}^{(t)})_t$ is indeed a sequence of noisy subgradient oracles for $\big(f, (x^{(t)})_t\big)$.

Theorem 3 *The sequences* $(\widetilde{g}^{(t)})_t$ *and* $(x^{(t)})_t$ *in Algorithm 3 satisfy* $\|\mathbb{E}\big[\widetilde{g}^{(t)} \bigm| x^{(t)}\big] - g(x^{(t)})\|_1 \leq \epsilon_0 + 2\epsilon_1$*,* $\|\widetilde{g}^{(t)}\|_2 \leq 18$ *and* $x^{(t+1)} = \operatorname{argmin}_{x \in [0,1]^n} \|x - (x^{(t)} - \eta \widetilde{g}^{(t)})\|_2$*.*

Proof. Fix t and $\tau = (t \bmod T)$. According to lines 4 and 5 of the algorithm, we have

$$\begin{cases} \widetilde{g}^{(t)} = \widetilde{g}^{(t,0)} & \text{if } \tau = 0 \\ \widetilde{g}^{(t)} = \widetilde{g}^{(t-\tau,\tau)} + \widetilde{d}_+^{(t)} + \widetilde{d}_-^{(t)} & \text{otherwise.} \end{cases}$$

We first study the expectation of the term $\widetilde{g}^{(t-\tau,\tau)}$, which is generated by the $\mathsf{GSample}$ procedure. Using the law of total expectation, it satisfies

$$\begin{aligned} \mathbb{E}[\widetilde{g}^{(t-\tau,\tau)} \mid x^{(t)}] &= \mathbb{E}\Big[\mathbb{E}[\widetilde{g}^{(t-\tau,\tau)} \mid (\widetilde{g}^{(t-\tau,k)})_{k<\tau}, x^{(t-\tau)}, x^{(t)}] \Bigm| x^{(t)}\Big] \\ &= \mathbb{E}\Big[\mathbb{E}[\widetilde{g}^{(t-\tau,\tau)} \mid (\widetilde{g}^{(t-\tau,k)})_{k<\tau}, x^{(t-\tau)}] \Bigm| x^{(t)}\Big] \end{aligned}$$

since $x^{(t)}$ does not convey any information about the output of $\mathsf{GSample}(x^{(t-\tau)}, T)$ when $(\widetilde{g}^{(t-\tau,k)})_{k<\tau}$ and $x^{(t-\tau)}$ are known. Consequently,

$$\begin{aligned} \Big\|\mathbb{E}[\widetilde{g}^{(t-\tau,\tau)} - g(x^{(t-\tau)}) \mid x^{(t)}]\Big\|_1 &\leq \mathbb{E}\Big[\Big\|\mathbb{E}[\widetilde{g}^{(t-\tau,\tau)} \mid (\widetilde{g}^{(t-\tau,k)})_{k<\tau}, x^{(t-\tau)}] - g(x^{(t-\tau)})\Big\|_1 \Bigm| x^{(t)}\Big] \\ &\leq \epsilon_0 \end{aligned}$$

using the triangle inequality and Assumption 1.

We now study the expectation of the term $\widetilde{d}_+^{(t)} + \widetilde{d}_-^{(t)}$ generated by the $\mathsf{GDSample}$ procedure

Algorithm 3 Subgradient descent algorithm for the Lovász extension f.

Input: two integers $0 < T < N$, two reals $\epsilon_0, \epsilon_1 > 0$.
Output: point $\bar{x} \in [0,1]^n$.

```
Set $x^{(0)} = 0^n \in [0,1]^n$.
for $t = 0, \dots, N$ do
    Set $\tau = (t \bmod T)$.
    ▷ Computation of the subgradient estimate $\widetilde{g}^{(t)}$:
    If $\tau = 0$: sample $\widetilde{g}^{(t,0)}, \dots, \widetilde{g}^{(t,T-1)}$ using $\mathsf{GSample}(x^{(t)}, T, \epsilon_0)$. Set $\widetilde{g}^{(t)} = \widetilde{g}^{(t,0)}$.
    If $\tau \neq 0$: sample $\widetilde{d}_+^{(t)}$ using $\mathsf{GDSample}\big(x^{(t-\tau)}, e_+^{(t-1)}, \epsilon_1\big)$ and sample $\widetilde{d}_-^{(t)}$ using $\mathsf{GDSample}\big(x^{(t-\tau)} + e_+^{(t-1)}, e_-^{(t-1)}, \epsilon_1\big)$. Set $\widetilde{g}^{(t)} = \widetilde{g}^{(t-\tau,\tau)} + \widetilde{d}_+^{(t)} + \widetilde{d}_-^{(t)}$.
    ▷ Update of the position to $x^{(t+1)}$:
    Compute $x^{(t+1)} = \mathrm{argmin}_{x \in [0,1]^n} \|x - (x^{(t)} - \eta \widetilde{g}^{(t)})\|_2$, that is $x^{(t+1)} = x^{(t)} + u^{(t)}$ where
        $u_i^{(t)} = \begin{cases} -x_i^{(t)} & \text{if } \eta \widetilde{g}_i^{(t)} > x_i^{(t)} \\ 1 - x_i^{(t)} & \text{if } \eta \widetilde{g}_i^{(t)} < -(1 - x_i^{(t)}) \\ -\eta \widetilde{g}_i^{(t)} & \text{otherwise} \end{cases}$
    for each $i \in [n]$, and $\eta = \sqrt{\frac{n}{18^2 N}}$.
    ▷ Update of the difference to $e^{(t)} = x^{(t+1)} - x^{(t-\tau)}$:
    If $\tau = 0$, set $e^{(t)} = u^{(t)}$.
    If $\tau \neq 0$, set $e^{(t)} = e^{(t-1)} + u^{(t)}$.
Output $\bar{x} = \frac{1}{N} \sum_{t=0}^{N-1} x^{(t)}$.
```

when $\tau \neq 0$. We have

$$\begin{aligned}\mathbb{E}[\widetilde{d}_+^{(t)} + \widetilde{d}_-^{(t)} \mid x^{(t)}] &= \mathbb{E}\Big[\mathbb{E}[\widetilde{d}_+^{(t)} \mid x^{(t-\tau)}, e_+^{(t-1)}, x^{(t)}] + \mathbb{E}[\widetilde{d}_-^{(t)} \mid x^{(t-\tau)} + e_+^{(t-1)}, e_-^{(t-1)}, x^{(t)}] \;\Big|\; x^{(t)}\Big] \\ &= \mathbb{E}\Big[\mathbb{E}[\widetilde{d}_+^{(t)} \mid x^{(t-\tau)}, e_+^{(t-1)}] + \mathbb{E}[\widetilde{d}_-^{(t)} \mid x^{(t-\tau)} + e_+^{(t-1)}, e_-^{(t-1)}] \;\Big|\; x^{(t)}\Big]\end{aligned}$$

where the first line is by the law of total expectation, and the second line is by independence between random variables. Moreover, according to Assumption 2, $\|\mathbb{E}[\widetilde{d}_+^{(t)} \mid x^{(t-\tau)}, e_+^{(t-1)}] - (g(x^{(t-\tau)} + e_+^{(t-1)}) - g(x^{(t-\tau)}))\|_1 \leq \epsilon_1$ and $\|\mathbb{E}[\widetilde{d}_-^{(t)} \mid x^{(t-\tau)} + e_+^{(t-1)}, e_-^{(t-1)}] - (g(x^{(t)}) - g(x^{(t-\tau)} + e_+^{(t-1)}))\|_1 \leq \epsilon_1$ (where we used that $x^{(t)} = x^{(t-\tau)} + e_+^{(t-1)} + e_-^{(t-1)}$). Thus, by the triangle inequality,

$$\|\mathbb{E}[\widetilde{d}_+^{(t)} + \widetilde{d}_-^{(t)} - (g(x^{(t)}) - g(x^{(t-\tau)})) \mid x^{(t)}]\|_1 \leq 2\epsilon_1.$$

This concludes the proof of the first part of the theorem since $\|\mathbb{E}[\widetilde{g}^{(t)} \mid x^{(t)}] - g(x^{(t)})\|_1 = \|\mathbb{E}[\widetilde{g}^{(t,0)} - g(x^{(t)}) \mid x^{(t)}]\|_1 \leq \epsilon_0$ when $\tau = 0$, and $\|\mathbb{E}[\widetilde{g}^{(t)} \mid x^{(t)}] - g(x^{(t)})\|_1 \leq \|\mathbb{E}[\widetilde{g}^{(t-\tau,\tau)} - g(x^{(t-\tau)}) \mid x^{(t)}]\|_1 + \|\mathbb{E}[\widetilde{d}_+^{(t)} + \widetilde{d}_-^{(t)} - (g(x^{(t)}) - g(x^{(t-\tau)})) \mid x^{(t)}]\|_1 \leq \epsilon_0 + 2\epsilon_1$ when $\tau \neq 0$. The second part of the theorem is a direct application of the triangle inequality using that $\|\widetilde{g}^{(t-\tau,\tau)}\|_2 \leq 4$ and $\|\widetilde{d}_+^{(t)}\|_2, \|\widetilde{d}_-^{(t)}\|_2 \leq 7$ (Assumptions 1 and 2). The last part of the theorem is line 6 of the algorithm. □

The above result shows that Algorithm 3 is a (noisy) subgradient descent for the Lovász extension. Consequently, the result of Proposition 3 can be applied to the output $\bar{x}$ of the algorithm. Since we aim for a subquadratic running time in n, we must update the vectors $x^{(t)}$, $\widetilde{g}^{(t)}$, $u^{(t)}$ and $e^{(t)}$ in time less than their dimension. Here, we do not discuss the data structure used for this purpose (see Section 5.1). Nevertheless, we recall that the outputs of GSample and GDSample are 1-sparse, thus most of the coordinates do not change between two consecutive steps.

Fact 1 *At step t of the algorithm: $\widetilde{g}^{(t)}$ and $u^{(t)}$ are 3-sparse, $e_+^{(t)}$ and $e_-^{(t)}$ are $3(\tau+1)$-sparse, if $\tau \neq 0$ then $e_+^{(t-1)}$ and $e_+^{(t)}$ (resp. $e_-^{(t-1)}$ and $e_-^{(t)}$) can differ only at positions where $u^{(t)}$ is non-zero.*

Corollary 1 *The output $\bar{x}$ of Algorithm 3 satisfies $\mathbb{E}[f(\bar{x})] \leq \min_x f(x) + 18\sqrt{n/N} + \epsilon_0 + 2\epsilon_1$. The total run-time of steps 2.(b) and 2.(c) is $O\Big(\frac{N}{T}\Big(c_{\mathsf{GS}}(T,\epsilon_0) + \sum_{\tau=1}^{T} c_{\mathsf{GDS}}(3\tau,\epsilon_1)\Big)\Big)$.*

Proof. According to Theorem 3, $(\widetilde{g}^{(t)})_t$ is a sequence of ϵ-noisy subgradient oracles for the Lovász extension f, where $\epsilon = \epsilon_0 + 2\epsilon_1$ and $(x^{(t)})_t$ obeys the subgradient descent update rule $x^{(t+1)} = \operatorname{argmin}_{x \in [0,1]^n} \|x - (x^{(t)} - \eta\widetilde{g}^{(t)})\|_2$. Moreover, $\|x - x^\star\|_2 \leq \sqrt{n}$, $\|x - x^\star\|_\infty \leq 1$ for all $x \in [0,1]^n$, and $\|\widetilde{g}^{(t)}\|_2 \leq 18$. Consequently, we obtain from Proposition 3 that $\mathbb{E}[f(\bar{x})] \leq f(x^\star) + 18\sqrt{n/N} + \epsilon_0 + 2\epsilon_1$, where we used the step size parameter $\eta = \sqrt{\frac{n}{18^2 N}}$. The time complexity of steps 2.(b) and 2.(c) is a direct consequence of Assumptions 1 and 2 and Fact 1. □

5 Subquadratic Approximate Submodular Minimization

We construct two classical (Section 5.2) and two quantum (Section 5.3) procedures satisfying the Assumptions 1 and 2 described in the previous section. These procedures will require a particular data structure, strongly inspired from the work of [2], that is maintained throughout the subgradient descent algorithm at negligible cost (Section 5.1). Our final algorithms solve the approximate submodular minimization problem in time $\widetilde{O}(n^{3/2}/\epsilon^2 \cdot \mathrm{EO})$ classically, and $\widetilde{O}(n^{5/4}/\epsilon^{5/2} \cdot \log(1/\epsilon) \cdot \mathrm{EO})$ quantumly.

5.1 Data structures and k-Covers

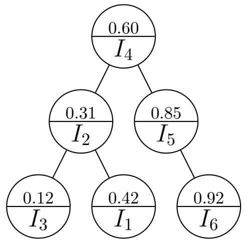
We describe two data structures needed to implement $\mathsf{GSample}$ and $\mathsf{GDSample}$ respectively, and we establish their main properties. The first data structure $\mathsf{D}(x)$ contains standard information about the input point x of $\mathsf{GSample}(x, T, \epsilon)$. It is defined as follows.

Definition 5 *Given $x \in \mathbb{R}^n$ and the permutation P consistent with x, we define $\mathsf{D}(x) = (L_x, A_x, T_x)$ to be the data structure made of the following elements: a doubly linked list L_x storing P, an array A_x storing at position $i \in [n]$ the value x_i with a pointer to the corresponding entry in P, and a self-balancing binary search tree T_x (e.g. red-black tree [48]) with a node for each $i \in [n]$ keyed by the value x_i and containing the size of its subtree.*

The second data structure $\mathsf{D}(x, y, \mathcal{I})$ is based on a property of submodular functions established in [2] that requires the following definition of a k-cover.

Definition 6 *Consider $x, y \in [0,1]^n$ and let P and Q be the permutations consistent with x and y respectively. We say that a partition $\mathcal{I} = \{I_1, \dots, I_k\}$ of $[n]$ is a* k-cover *of (x, y) if, for each $s \in [k]$, the preimage of I_s under both P and Q is a set of consecutive numbers, and $x_i = y_i$ for all $i \in I_s$ if $|I_s| > 1$.*

	1	2	3	4	5	6	7	8	9	10
x	0.85	0.58	**0.42**	0.53	0.60	0.78	**0.12**	0.27	0.92	0.31
y	0.85	0.58	**0.65**	0.53	0.60	0.78	**0.90**	0.27	0.92	0.31
P	9	1	6	5	2	4	③	10	8	⑦
Q	9	⑦	1	6	③	5	2	4	10	8



$$I_1 = \{3\},\ I_2 = \{10, 8\},\ I_3 = \{7\},\ I_4 = \{5, 2, 4\},\ I_5 = \{1, 6\},\ I_6 = \{9\}.$$

Fig. 1. An illustration of a 6-cover $\mathcal{I} = \{I_1, \dots, I_6\}$ for some $x, y \in [0,1]^{10}$ and their corresponding permutations P, Q. The circled numbers correspond to the positions where x and y differ (these values must belong to singletons in the cover). The binary tree corresponds to $T_x^{\mathcal{I}}$ in $\mathsf{D}(x, y, \mathcal{I})$.

An example of a 6-cover is given in Figure 5.1. Observe that there always exists a cover of size at most $3k+1$ if the difference $e = x - y$ is k-sparse. Our implementations of $\mathsf{GDSample}(x, e, \epsilon)$ will require to store a cover of $(x, x+e)$ approaching that size. Before explaining the reasons why a cover is useful, we describe the data structure $\mathsf{D}(x, y, \mathcal{I})$.

Definition 7 *Given $x, y \in \mathbb{R}^n$ and a k-cover $\mathcal{I} = \{I_1, \dots, I_k\}$ of (x, y), we define $\mathsf{D}(x, y, \mathcal{I}) = \big(\mathsf{D}(x), \mathsf{D}(y), A_x^{\mathcal{I}}, A_y^{\mathcal{I}}, T_x^{\mathcal{I}}, T_y^{\mathcal{I}}\big)$ to be the data structure made of the following elements: $\mathsf{D}(x)$ and $\mathsf{D}(y)$ (described in Definition 5), two dynamic arrays $A_x^{\mathcal{I}}$ and $A_y^{\mathcal{I}}$ of size k storing at position $s \in [k]$ the pairs $(\mathrm{argmax}_{i \in I_s} x_i, \mathrm{argmin}_{i \in I_s} x_i)$ and $(\mathrm{argmax}_{i \in I_s} y_i, \mathrm{argmin}_{i \in I_s} y_i)$ respectively, two self-balancing binary search trees $T_x^{\mathcal{I}}$ and $T_y^{\mathcal{I}}$ with a node for each $s \in [k]$ keyed by the value of $\max_{i \in I_s} x_i$ and $\max_{i \in I_s} y_i$ respectively.*

The next lemma is the crucial property established in [2] about k-covers. It shows that the coordinates $g(y)_i - g(x)_i$ of the subgradient difference have constant sign over any set

I_s of the cover when $y \geq x$ or $y \leq x$. In particular, the ℓ_1-norm $\|g(y)_{I_s} - g(x)_{I_s}\|_1$ can be deduced from the value of $\sum_{i \in I_s} g(y)_i - g(x)_i$ (we recall that $g(y)_{I_s} - g(x)_{I_s}$ is the subvector of $g(y) - g(x)$ made of the values at positions $i \in I_s$).

Lemma 4 ([2]) *Consider $x, y \in [0,1]^n$ such that $y \geq x$ or $y \leq x$, and let $\{I_1, \ldots, I_k\}$ be a k-cover of (x, y). Then, for each $s \in [k]$, the coordinates $g(y)_i - g(x)_i$ have the same sign for all $i \in I_s$. In particular, $|\sum_{i \in I_s} g(y)_i - g(x)_i| = \|g(y)_{I_s} - g(x)_{I_s}\|_1$.*

Proof. Let P and Q denote the permutations consistent with x and y respectively. Consider $s \in [k]$ such that $|I_s| > 1$ (the result is trivial when $|I_s| = 1$). By definition of a k-cover, there exist three integers a_s, a'_s, ℓ_s such that $I_s = \{P_{a_s}, P_{a_s+1}, \ldots, P_{a_s+\ell_s}\} = \{Q_{a'_s}, Q_{a'_s+1}, \ldots, Q_{a'_s+\ell_s}\}$. Assume that $y \geq x$ (the case $y \leq x$ is symmetric). Since $x_i = y_i$ for all $i \in I_s$, we must have $P[a_s - 1] \subseteq Q[a'_s - 1]$. Thus, by the diminishing returns property of submodular functions, $F(P[a_s + \ell]) - F(P[a_s + \ell - 1]) \geq F(Q[a'_s + \ell]) - F(Q[a'_s + \ell - 1])$ for all $0 \leq \ell \leq \ell_s$. We conclude that $g(y)_i - g(x)_i \leq 0$ for all $i \in I_s$. □

Note that the condition $y \geq x$ or $y \leq x$ is crucial in the above result, which is why we impose $e \geq 0$ or $e \leq 0$ in Assumption 2. We now describe three useful operations that can be handled in logarithmic time using $\mathsf{D}(x, y, \mathcal{I})$ and the above lemma. The first two operations originate from the work of [2] (the proofs are given for completeness), whereas the third one is new to this work (in [2], the authors recompute the cover entirely at each update).

Proposition 4 *Consider $x, y \in \mathbb{R}^n$ such that $x \geq y$ or $x \leq y$, and let $\mathcal{I} = \{I_1, \ldots, I_k\}$ be a k-cover of (x, y). Then, using $\mathsf{D}(x, y, \mathcal{I})$, the following operations can be handled in time $O(\log(n) + \mathrm{EO})$, $O(\log(n) \cdot \mathrm{EO})$ and $O(\log n)$ respectively:*

- (Subnorm) *Given $s \in [k]$, output $\|g(y)_{I_s} - g(x)_{I_s}\|_1$.*
- (Subsampling) *Given $s \in [k]$, sample $i \sim \mathcal{D}_{g(y)_{I_s} - g(x)_{I_s}}$.*
- (Update) *Given a 1-sparse vector $e \in \mathbb{R}^n$, update the data structure to $\mathsf{D}(x + e, y, \mathcal{I}')$ where $\mathcal{I}'$ is a cover of $(x + e, y)$ of size at most $k + 3$.*

Proof. Let P be the permutation consistent with x. Observe that the rank P_i^{-1} of any x_i can be computed in $\log(n)$ time using T_x (since each node in the tree contains the size of its subtree).

(Subnorm) By definition of a k-cover, there exist $a_s \leq b_s$ such that $I_s = \{P_{a_s}, P_{a_s+1}, \ldots, P_{b_s}\}$. Thus, $\sum_{i \in I_s} g(x)_i = \sum_{i=a_s}^{b_s} F(P[i]) - F(P[i-1]) = F(P[b_s]) - F(P[a_s - 1])$. Since a_s and b_s can be obtained in time $O(\log n)$ using $\mathsf{D}(x, y, \mathcal{I})$, this sum can be computed in time $O(\log(n) + \mathrm{EO})$, and similarly for $\sum_{i \in I_s} g(y)_i$. According to Lemma 4, the difference $|\sum_{i \in I_s} g(y)_i - g(x)_i|$ is equal to the ℓ_1-norm of $g(y)_{I_s} - g(x)_{I_s}$.

(Subsampling) We find the highest node $i_h \in I_s$ in the tree T_x, and we compute its rank $c_s = P_{i_h}^{-1}$ in time $O(\log n)$. It partitions I_s into three sets $A = \{P_{a_s}, P_{a_s+1}, \ldots, P_{c_s-1}\}$, $B = \{P_{c_s}\}$ and $C = \{P_{c_s+1}, \ldots, P_{b_s}\}$. We compute the ℓ_1-norm of $g(y) - g(x)$ restricted to each of these sets in time $O(\mathrm{EO})$, and we sample A, B or C with probability $\frac{\|g(y)_A - g(x)_A\|_1}{\|g(y)_{I_s} - g(x)_{I_s}\|_1}$, $\frac{\|g(y)_B - g(x)_B\|_1}{\|g(y)_{I_s} - g(x)_{I_s}\|_1}$ and $\frac{\|g(y)_C - g(x)_C\|_1}{\|g(y)_{I_s} - g(x)_{I_s}\|_1}$ respectively. If the set we obtain is a singleton, we terminate and output the value it contains, otherwise we continue recursively to sample in the corresponding subtree of T_x.

(Update) We explain how to update the k-cover (the other parts of the data structure being standard to update). Let i be the position where $e_i \neq 0$, and denote $r = P_i^{-1}$ the

rank of i in P before the update. First, split the set $I_s = \{P_{a_s}, \dots, P_{r-1}, i, P_{r+1}, \dots, P_{b_s}\}$ containing i into three parts $\{P_{a_s}, P_{a_s+1}, P_{r-1}\}$, $\{i\}$ and $\{P_{r+1}, \dots, P_{b_s}\}$. Then, identify the rank c such that $x_{P_c} > x_i + e_i > x_{P_{c+1}}$ and split the set containing P_c into two parts. These operations can be done in $O(\log n)$ time. The size of the cover is increased by at most 3. □

Finally, observe that we have to maintain two instances of $\mathsf{D}(x, y, \mathcal{I})$ in Algorithm 3, one corresponding to the pair $(x^{(t-\tau)}, x^{(t-\tau)} + e_+^{(t-1)})$ (needed for $\widetilde{d}_+^{(t)}$), and the other one corresponding to $(x^{(t-\tau)} + e_+^{(t-1)}, x^{(t)})$ (needed for $\widetilde{d}_-^{(t)}$). The total update cost is negligible because of Fact 1.

Corollary 2 *One can maintain throughout Algorithm 3 two data structures* $\mathsf{D}(x^{(t-\tau)}, x^{(t-\tau)} + e_+^{(t-1)}, \mathcal{I})$ *and* $\mathsf{D}(x^{(t-\tau)} + e_+^{(t-1)}, x^{(t)}, \mathcal{I}')$, *where* $\mathcal{I}$ *and* $\mathcal{I}'$ *are covers of size at most* 9τ *and* 27τ *respectively. The update time, at each step of the algorithm, is* $O(\log n)$.

Proof. This is a direct consequence of Fact 1 and Proposition 4. The update from $e_+^{(t-1)}$ to $e_+^{(t)}$ (resp. $e_-^{(t-1)}$ to $e_-^{(t)}$) is 3-sparse, thus each step increases the size of the cover associated with $(x^{(t-\tau)}, x^{(t-\tau)} + e_+^{(t-1)})$ by 9, and the size of the cover associated with $(x^{(t-\tau)} + e_+^{(t-1)}, x^{(t)}) = (x^{(t-\tau)} + e_+^{(t-1)}, x^{(t-\tau)} + e_+^{(t-1)} + e_-^{(t-1)})$ by 27. When $\tau = 0$, the sizes are reset to at most 9 and 27 respectively. □

5.2 Classical Approximate Submodular Minimization

We conclude the part on classical approximate submodular minimization by describing the two procedures $\mathcal{C}$-GS and $\mathcal{C}$-GDS for $\mathsf{GSample}$ and $\mathsf{GDSample}$ respectively that lead to an $\widetilde{O}(n^{3/2}/\epsilon^2 \cdot \mathrm{EO})$ algorithm. Both are based on the following unbiased estimator X_u of any vector $u \in \mathbb{R}^n$.

Fact 2 *Given a non-zero vector* $u \in \mathbb{R}^n$, *consider the vector-valued random variable* X_u *that equals* $\|u\|_1 \operatorname{sgn}(u_i) \cdot \vec{1}_i$ *with probability* $\frac{|u_i|}{\|u\|_1}$. *Then,* $\mathbb{E}[X_u] = u$ *and* $\|X_u\|_2 = \|u\|_1$.

In the case of gradient sampling (Assumption 1), we construct T samples from $X_{g(x)}$ by using the sampling algorithm of Lemma 1. In the case of gradient difference sampling (Assumption 2), we construct one sample from $X_{g(x+e)-g(x)}$ by using the subnorm and subsampling operations of Proposition 4. These two procedures are described in Algorithms 4 and 5 respectively.

Algorithm 4 Classical Gradient Sampling ($\mathcal{C}$-GS).

```
Input: $x \in [0,1]^n$ stored in $\mathsf{D}(x)$, an integer $T$.
Output: a sequence of estimates $\widetilde{g}^1, \dots, \widetilde{g}^T$ of $g(x)$.
1: Compute $\|g(x)\|_1$ and sample $i_1, \dots, i_T \sim \mathcal{D}_{g(x)}$ using Lemma 1.
2: For each $j \in [T]$, compute $g(x)_{i_j}$ and output $\widetilde{g}^j = \|g(x)\|_1 \operatorname{sgn}(g(x)_{i_j}) \cdot \vec{1}_{i_j}$.
```

Proposition 5 *The classical procedure* $\mathcal{C}$-$\mathsf{GS}(x, T)$ *of Algorithm 4 satisfies the conditions given on* $\mathsf{GSample}(x, T, 0)$ *in Assumption 1, with time complexity* $c_{\mathsf{GS}}(T, 0) = O((n + T) \cdot \mathrm{EO})$.

Proof. By Fact 2, we have $\mathbb{E}[\widetilde{g}^j \mid \widetilde{g}^1, \dots, \widetilde{g}^{j-1}, x] = \mathbb{E}[\widetilde{g}^j \mid x] = g(x)$ and $\|\widetilde{g}^j\|_2 = \|g(x)\|_1 \leq 3$. Line 1 takes time $O(n \cdot \mathrm{EO} + T)$, and line 2 takes time $O(T \cdot \mathrm{EO})$. □

Proposition 6 *The classical procedure* $\mathcal{C}$-$\mathsf{GDS}(x, e)$ *of Algorithm 5 satisfies the conditions given on* $\mathsf{GDSample}(x, e, 0)$ *in Assumption 2, with time complexity* $c_{\mathsf{GDS}}(k, 0) = \widetilde{O}(k \cdot \mathrm{EO})$.

Algorithm 5 Classical Gradient Difference Sampling ($\mathcal{C}$-GDS).

```
Input: $x, x+e \in [0,1]^n$ and a cover $\mathcal{I} = \{I_1, \dots, I_{k'}\}$ of $(x, x+e)$ stored in $\mathsf{D}(x, x+e, \mathcal{I})$, where $e$ is $k$-sparse and $k' \leq 9k$.
Output: an estimate $\widetilde{d}$ of $g(x+e) - g(x)$.
Let $u = (\|g(x+e)_{I_s} - g(x)_{I_s}\|_1)_{s \in [k']} \in \mathbb{R}^{k'}$.
1: Compute $\|u\|_1$ and sample $s \sim \mathcal{D}_u$ using Lemma 1 and the subnorm operation of Proposition 4.
2: Sample $i \sim \mathcal{D}_{g(x+e)_{I_s} - g(x)_{I_s}}$ using the subsampling operation of Proposition 4.
3: Compute $g(x+e)_i - g(x)_i$ and output $\widetilde{d} = \|u\|_1 \operatorname{sgn}(g(x+e)_i - g(x)_i) \cdot \vec{1}_i$.
```

Proof. The value i is distributed according to $\frac{\|g(x+e)_{I_s} - g(x)_{I_s}\|_1}{\|u\|_1} \cdot \frac{|g(x+e)_i - g(x)_i|}{\|g(x+e)_{I_s} - g(x)_{I_s}\|_1} = \frac{|g(x+e)_i - g(x)_i|}{\|g(x+e) - g(x)\|_1}$ (since $\|u\|_1 = \|g(x+e) - g(x)\|_1$). This corresponds to $\mathcal{D}_{g(x+e)-g(x)}$. Consequently, using Fact 2, $\mathbb{E}[\widetilde{d} \mid x, e] = g(x+e) - g(x)$ and $\|\widetilde{d}\|_2 = \|g(x+e) - g(x)\|_1 \leq 6$. Line 1 takes time $O(k(\log(n) + \mathrm{EO}))$, line 2 takes time $O(\log(n) \cdot \mathrm{EO})$ and line 3 takes time $O(\mathrm{EO})$. □

Finally, we analyze the cost of using the two above procedures in the subgradient descent algorithm studied in Section 4.

Theorem 4 *There is a classical algorithm that, given a submodular function $F : 2^V \to [-1,1]$ and $\epsilon > 0$, computes a set $\bar{S}$ such that $\mathbb{E}[F(\bar{S})] \leq \min_{S \subseteq V} F(S) + \epsilon$ in time $\widetilde{O}(n^{3/2}/\epsilon^2 \cdot \mathrm{EO})$.*

Proof. We instantiate Algorithm 3 with the procedures $\mathcal{C}$-GS and $\mathcal{C}$-GDS of Algorithms 4 and 5 respectively, and we choose the input parameters $T = \sqrt{n}$, $N = 18^2 n/\epsilon^2$ and $\epsilon_0 = \epsilon_1 = 0$. The data structures needed for $\mathcal{C}$-GS and $\mathcal{C}$-GDS can be updated in time $O(\log n)$ per step (Corollary 2). According to Corollary 1, we obtain an output $\bar{x}$ such that $\mathbb{E}[f(\bar{x})] \leq \min_x f(x) + \epsilon$ in time $\widetilde{O}\left(\frac{\sqrt{n}}{\epsilon^2}\left(n + \sum_{\tau=1}^{\sqrt{n}} \tau\right) \cdot \mathrm{EO}\right) = \widetilde{O}(n^{3/2}/\epsilon^2 \cdot \mathrm{EO})$. Finally, using Proposition 1, we can convert $\bar{x}$ into a set $\bar{S} \subseteq V$ such that $\mathbb{E}[F(\bar{S})] \leq \min_{S \subseteq V} F(S) + \epsilon$ in time $O(n \log n + n \cdot \mathrm{EO})$. □

5.3 Quantum Approximate Submodular Minimization

We conclude the part on quantum approximate submodular minimization by describing the two procedures $\mathcal{Q}$-GS and $\mathcal{Q}$-GDS for $\mathsf{GSample}$ and $\mathsf{GDSample}$ respectively that lead to an $\widetilde{O}(n^{5/4}/\epsilon^{5/2} \cdot \log(1/\epsilon) \cdot \mathrm{EO})$ algorithm. Both are based on the following noisy estimator Y_u of any vector $u \in \mathbb{R}^n$.

Fact 3 *Given a non-zero vector $u \in \mathbb{R}^n$, a real $\Gamma > 0$ and a set $S \subseteq [n]$ such that $\Gamma \geq \|u_S\|_1$, consider the vector-valued random variable Y_u that equals $\Gamma \operatorname{sgn}(u_i) \cdot \vec{1}_i$ where $i \sim \mathcal{D}_u(\Gamma, S)$. Then, $\|\mathbb{E}[Y_u] - u\|_1 = |\Gamma - \|u\|_1|$ and $\|Y_u\|_2 = \Gamma$.*

In the case of gradient sampling (Assumption 1), we construct T samples from $Y_{g(x)}$ by using the sampling algorithm of Theorem 1. In the case of gradient difference sampling (Assumption 2), we construct one sample from $Y_{g(x+e)-g(x)}$ by using Lemma 2 and the following evaluation oracle derived from Proposition 4.

Proposition 7 *There is a quantum algorithm, represented as a unitary operator $\mathcal{O}$, that given $x, y \in [0,1]^n$ and a k-cover $\mathcal{I} = \{I_1, \dots, I_k\}$ of (x, y) stored in $\mathsf{D}(x, y, \mathcal{I})$, satisfies*

$\mathcal{O}(|s\rangle|0\rangle) = |s\rangle \big| \|g(y)_{I_s} - g(x)_{I_s}\|_1 \big\rangle$, for all $s \in [k]$. The time complexity of this algorithm is $O(\log(n) + \mathrm{EO})$.

The procedures GSample and GDSample are described in Algorithms 6 and 7 respectively. There is a non-zero probability that the computation of the setup parameters is incorrect. In this case, we cannot guarantee that Assumptions 1 and 2 are satisfied. Fortunately, the dependence of the time complexity on the inverse failure probability is logarithmic. Thus, it will not impact the analysis significantly.

Algorithm 6 Quantum Gradient Sampling ($\mathcal{Q}$-GS).

```
Input: $x \in [0,1]^n$ stored in $\mathsf{D}(x)$, an integer $T$, two reals $0 < \epsilon, \delta < 1$.
Output: a sequence of estimates $\widetilde{g}^1, \ldots, \widetilde{g}^T$ of $g(x)$.
1: Compute the setup parameters $(\Gamma, S, M)$ using Proposition 2 with input $g(x)$, $T$, $\epsilon/3$, $\delta$.
2: Sample $i^1, \ldots, i^T \sim \mathcal{D}_{g(x)}(\Gamma, S)$ using Theorem 1 with input $g(x)$, $T$, $(\Gamma, S, M)$.
3: For each $j \in [T]$, compute $g(x)_{i_j}$ and output $\widetilde{g}^j = \Gamma \operatorname{sgn}(g(x)_{i_j}) \cdot \vec{1}_{i_j}$.
```

Proposition 8 *The quantum procedure $\mathcal{Q}$-$\mathsf{GS}(x, T, \epsilon, \delta)$ of Algorithm 6 satisfies the conditions given on $\mathsf{GSample}(x, T, \epsilon)$ in Assumption 1 with probability $1 - \delta$, and time complexity $c_{\mathsf{GDS}}(T, \epsilon) = O((\sqrt{nT} + \sqrt{n}/\epsilon) \log(1/\delta) \cdot \mathrm{EO})$.*

Proof. Let us denote $\mathbf{V}$ the event that the setup parameters (Γ, S, M) computed at line 1 of the algorithm are valid (i.e. they satisfy the properties given in Proposition 2). We have $\Pr[\mathbf{V}] \geq 1 - \delta$. According to Theorem 1, if $\mathbf{V}$ holds, then $i^1, \ldots, i^T$ are T independent samples from $\mathcal{D}_{g(x)}(\Gamma, S)$. Consequently, using Fact 3, we have $\|\mathbb{E}[\widetilde{g}^j \mid x, (\Gamma, S, M), \mathbf{V}] - g(x)\|_1 = |\Gamma - \|g(x)\|_1| \leq (\epsilon/3)\|g(x)\|_1 \leq \epsilon$ and $\|\widetilde{g}^j\|_2 \leq (1 + \epsilon/3)\|g(x)\|_1 \leq 4$. Moreover, since $\widetilde{g}^1, \ldots, \widetilde{g}^T$ are independent *conditioned on* (Γ, S, M), by the law of total expectation $\|\mathbb{E}[\widetilde{g}^j \mid x, \widetilde{g}^1, \ldots, \widetilde{g}^{j-1}, \mathbf{V}] - g(x)\|_1 = \big\|\mathbb{E}\big[\mathbb{E}[\widetilde{g}^j \mid x, (\Gamma, S, M), \mathbf{V}] \bigm| \widetilde{g}^1, \ldots, \widetilde{g}^{j-1}, \mathbf{V}\big] - g(x)\big\|_1 \leq \epsilon$. Finally, line 1 takes time $O((\sqrt{nT} + \sqrt{n}/\epsilon) \log(1/\delta) \cdot \mathrm{EO})$, line 2 takes time $O(\sqrt{nT} \cdot \mathrm{EO})$, and line 3 takes time $O(T \cdot \mathrm{EO})$. □

Algorithm 7 Quantum Gradient Difference Sampling ($\mathcal{Q}$-GDS).

```
Input: $x, x + e \in [0,1]^n$ and a cover $\mathcal{I} = \{I_1, \ldots, I_{k'}\}$ of $(x, x + e)$ stored in $\mathsf{D}(x, x + e, \mathcal{I})$,
    where $e$ is $k$-sparse and $k' \leq 9k$, two reals $0 < \epsilon, \delta < 1$.
Output: an estimate $\widetilde{d}$ of $g(x + e) - g(x)$.
Let $u = (\|g(x + e)_{I_s} - g(x)_{I_s}\|_1)_{s \in [k']} \in \mathbb{R}^{k'}$.
Compute $\|u\|_\infty$ with success probability $1 - \delta/2$, using Dürr-Høyer's algorithm [43] and
    Proposition 7. Denote the result by $M$.
Compute an estimate $\Gamma$ of $\|u\|_1$ with relative error $\epsilon/6$ and success probability $1 - \delta/2$,
    using Lemma 3 and Proposition 7.
Sample $s \sim \mathcal{D}_u$ using Lemma 2 on input $u$ and $M$.
Sample $i \sim \mathcal{D}_{g(x+e)_{I_s} - g(x)_{I_s}}$ using the subsampling operation of Proposition 4.
Compute $g(x + e)_i - g(x)_i$ and output $\widetilde{d} = \Gamma \operatorname{sgn}(g(x + e)_i - g(x)_i) \cdot \vec{1}_i$.
```

Proposition 9 *The quantum procedure $\mathcal{Q}$-$\mathsf{GDS}(x, e, \epsilon, \delta)$ of Algorithm 7 satisfies the conditions given on $\mathsf{GDSample}(x, e, \epsilon)$ in Assumption 2 with probability $1 - \delta$, and time complexity $c_{\mathsf{GS}}(k, \epsilon) = \widetilde{O}(\sqrt{k}/\epsilon \cdot \log(1/\delta) \cdot \mathrm{EO})$.*

Proof. Let us denote $\mathbf{V}$ the event that Γ and M are valid, i.e. $|\Gamma - \|g(x+e) - g(x)\|_1| \leq (\epsilon/6)\|g(x+e) - g(x)\|_1$ and $M = \|u\|_\infty$. We have $\Pr[\mathbf{V}] \geq 1 - \delta$. According to Lemma 2, if $\mathbf{V}$ holds, then s is sampled from $\mathcal{D}_u$. In this case, the value i computed at line 4 is distributed according to $\frac{\|g(x+e)_{I_s} - g(x)_{I_s}\|_1}{\|u\|_1} \cdot \frac{|g(x+e)_i - g(x)_i|}{\|g(x+e)_{I_s} - g(x)_{I_s}\|_1} = \frac{|g(x+e)_i - g(x)_i|}{\|g(x+e) - g(x)\|_1}$ (since $\|u\|_1 = \|g(x+e) - g(x)\|_1$). This corresponds to $\mathcal{D}_{g(x+e)-g(x)}$. Consequently, using Fact 3, $\|\mathbb{E}[\widetilde{d} \mid x, e, \Gamma, \mathbf{V}] - (g(x+e) - g(x))\|_1 = \left\| \frac{\Gamma}{\|g(x+e)-g(x)\|_1}(g(x+e) - g(x)) - (g(x+e) - g(x)) \right\|_1 = |\Gamma - \|g(x+e) - g(x)\|_1| \leq \epsilon/6\|g(x+e) - g(x)\|_1 \leq \epsilon$. Thus, $\|\mathbb{E}[\widetilde{d} \mid x, e, \mathbf{V}] - (g(x+e) - g(x))\|_1 \leq \epsilon$. Moreover, $\|\widetilde{d}\|_2 = \Gamma \leq (1 + \epsilon/6)\|g(x+e) - g(x)\|_1 \leq 7$. Finally, line 1 takes time $\widetilde{O}(\sqrt{k} \cdot \log(1/\delta) \cdot \mathrm{EO})$, line 2 takes time $\widetilde{O}(\sqrt{k}/\epsilon \cdot \log(1/\delta) \cdot \mathrm{EO})$, line 3 takes time $O(\sqrt{k} \cdot \mathrm{EO})$, and lines 4 and 5 take time $\widetilde{O}(\mathrm{EO})$. □

Finally, we analyze the cost of using the two above procedures in the subgradient descent algorithm studied in Section 4. Unlike in the previous section, the time complexities c_{GS} and c_{GDS} of $\mathcal{Q}$-GS and $\mathcal{Q}$-GDS depend on the accuracy ϵ. Consequently, it is more efficient to combine $\mathcal{Q}$-GS with the classical procedure $\mathcal{C}$-GDS when $n^{-1/2} \leq \epsilon \leq n^{-1/6}$, and to use Theorem 4 when $\epsilon \leq n^{-1/2}$.

Theorem 5 *There is a quantum algorithm that, given a submodular function* $F : 2^V \to [-1,1]$ *and* $\epsilon > 0$, *computes a set* $\bar{S}$ *such that* $\mathbb{E}[F(\bar{S})] \leq \min_{S \subseteq V} F(S) + \epsilon$ *in time* $\widetilde{O}(n^{5/4}/\epsilon^{5/2} \cdot \log(\frac{1}{\epsilon}) \cdot \mathrm{EO})$.

Proof. We distinguish two cases, depending on the value of ϵ. First, if $\epsilon \geq n^{-1/6}$, we instantiate Algorithm 3 with the quantum procedures $\mathcal{Q}$-GS and $\mathcal{Q}$-GDS of Algorithms 6 and 7 respectively. We choose the input parameters $T = \epsilon\sqrt{n}$, $N = 72^2 n/\epsilon^2$, $\epsilon_0 = \epsilon/4$, $\epsilon_1 = \epsilon/8$ and $\delta = \epsilon/(8N)$. According to Corollary 1 and Corollary 2, the run-time is $\widetilde{O}\Big(\frac{N}{T}\left(\sqrt{nT} + \frac{\sqrt{n}}{\epsilon} + \sum_{\tau=1}^{T} \frac{\sqrt{\tau}}{\epsilon}\right) \cdot \log(1/\delta) \cdot \mathrm{EO}\Big) = \widetilde{O}(n^{5/4}/\epsilon^{5/2} \cdot \log(1/\epsilon) \cdot \mathrm{EO})$. Let us denote $\mathbf{V}$ the event that *all* calls to $\mathcal{Q}$-GS and $\mathcal{Q}$-GDS in Algorithm 3 are correct (i.e. they satisfy Assumptions 1 and 2). By the union bound, $\Pr[\mathbf{V}] \geq 1 - 2N\delta \geq 1 - \epsilon/4$. Moreover, according to Corollary 1, the output $\bar{x}$ satisfies $\mathbb{E}[f(\bar{x}) \mid \mathbf{V}] \leq f(x^\star) + 3\epsilon/4$, where $x^\star \in \operatorname{argmin}_x f(x)$. Consequently, $\mathbb{E}[f(\bar{x})] = \Pr[\mathbf{V}] \cdot \mathbb{E}[f(\bar{x}) \mid \mathbf{V}] + (1 - \Pr[\mathbf{V}]) \cdot \mathbb{E}[f(\bar{x}) \mid \overline{\mathbf{V}}] \leq 1 \cdot (f(x^\star) + 3\epsilon/4) + \epsilon/4 \cdot 1 \leq f(x^\star) + \epsilon$. Finally, using Proposition 1, we can convert $\bar{x}$ into a set $\bar{S} \subseteq V$ such that $\mathbb{E}[F(\bar{S})] \leq \min_{S \subseteq V} F(S) + \epsilon$ in time $O(n \log n + n \cdot \mathrm{EO})$.

If $\epsilon \leq n^{-1/6}$, we instantiate Algorithm 3 with the quantum procedure $\mathcal{Q}$-GS of Algorithm 6 and the classical procedure $\mathcal{C}$-GDS of Algorithm 5. We choose the input parameters $T = n^{1/4}/\epsilon^{1/2}$, $N = 54^2 n/\epsilon^2$, $\epsilon_0 = \epsilon/3$, $\epsilon_1 = 0$ and $\delta = \epsilon/(3N)$. The run-time is $\widetilde{O}\Big(\frac{N}{T}\left(\left(\sqrt{nT} + \frac{\sqrt{n}}{\epsilon}\right)\log\frac{1}{\delta} + \sum_{\tau=1}^{T} \tau\right) \cdot \mathrm{EO}\Big) = \widetilde{O}(n^{5/4}/\epsilon^{5/2} \cdot \log(1/\epsilon) \cdot \mathrm{EO})$. The proof of correctness is similar to the above paragraph. □

Acknowledgements

The work of P.R., A.R. and M.S. was supported by the Singapore National Research Foundation, the Prime Minister's Office, Singapore and the Ministry of Education, Singapore under the Research Centres of Excellence programme under research grant R 710-000-012-135. The work of Y.H. and M.S. was supported by the QuantERA ERA-NET Cofund project QuantAlgo and the ANR project ANR-18-CE47-0010 QUDATA. This work was partly done while Y.H. was visiting CQT.

References

[1] Lee, Y.T., Sidford, A., Wong, S.C.: A faster cutting plane method and its implications for combinatorial and convex optimization. In: Proceedings of the 56th Symposium on Foundations of Computer Science (FOCS). pp. 1049–1065 (2015)

[2] Chakrabarty, D., Lee, Y.T., Sidford, A., Wong, S.C.: Subquadratic submodular function minimization. In: Proceedings of the 49th Symposium on Theory of Computing (STOC). pp. 1220–1231 (2017)

[3] Bach, F.: Learning with Submodular Functions: A Convex Optimization Perspective. Now Publishers (2013)

[4] Krause, A., Cevher, V.: Submodular dictionary selection for sparse representation. In: Proceedings of the 27th International Conference on Machine Learning (ICML). pp. 567–574 (2010)

[5] Nagano, K., Kawahara, Y., Aihara., K.: Size-constrained submodular minimization through minimum norm base. In: Proceedings of the 28th International Conference on Machine Learning (ICML). pp. 977–984 (2011)

[6] Iyer, R., Bilmes, J.A.: Submodular optimization with submodular cover and submodular knapsack constraints. In: Proceedings of the 26th Conference on Neural Information Processing Systems (NIPS). pp. 2436–2444 (2013)

[7] Queyranne, M., Schulz, A.: Scheduling unit jobs with compatible release dates on parallel machines with nonstationary speeds. In: Integer Programming and Combinatorial Optimization (IPCO). pp. 307–320 (1995)

[8] Narayanan, H.: Submodular Functions and Electrical Networks. North-Holland (2009)

[9] Hochbaum, D.: An efficient algorithm for image segmentation, markov random fields and related problems. Journal of the ACM 48(4), 686–701 (2001)

[10] Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(11), 1222–1239 (2001)

[11] Lin, H., Bilmes, J.A.: An application of the submodular principal partition to training data subset selection. In: Neural Information Processing Society (NIPS) Workshop (2010), nIPS Workshop on Discrete Optimization in Machine Learning: Submodularity, Sparsity & Polyhedra (DISCML)

[12] Lovász, L.: Submodular functions and convexity. Mathematical programming: The state of the art pp. 235–257 (1982)

[13] Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. Combinatorica 1(2), 169–197 (1981)

[14] Cunningham, W.: On submodular function minimization. Combinatorica 5(3), 185–192 (1985)

[15] Grötschel, M., Lovász, L., Schrijver, A.: Geometric algorithms and combinatorial optimization. Springer (1988)

[16] Schrijver, A.: A combinatorial algorithm minimizing submodular functions in strongly polynomial time. Journal of Combinatorial Theory, Series 80(2), 346–355 (2000)

[17] Iwata, S., Fleischer, L., Fujishige, S.: A combinatorial strongly polynomial algorithm for minimizing submodular functions. Journal of the ACM 48(4), 761–777 (2001)

[18] Fujishige, S.: Lexicographically optimal base of a polymatroid with respect to a weight vector. Mathematics of Operations Research 5(2), 186–196 (1980)

[19] Wolfe, P.: Finding the nearest point in a polytope. Mathematical Programming 11(1), 128–149 (1976)

[20] Aaronson, S.: Read the fine print. Nature Physics 11, 291–293 (2015)

[21] Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., Lloyd, S.: Quantum machine learning. Nature 549, 195–202 (2017)

[22] Ciliberto, C., Herbster, M., Ialongo, A.D., Pontil, M., Rocchetto, A., Severini, S., Wossnig, L.: Quantum machine learning: a classical perspective. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 474(2209), 20170551 (2018)

[23] Apeldoorn, J.v., Gilyén, A.: Improvements in quantum SDP-solving with applications. In: Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP). pp. 99:1–99:15 (2019)

[24] Arora, S., Kale, S.: A combinatorial, primal-dual approach to semidefinite programs. Journal of the ACM 63(2), 12:1–12:35 (2016)

[25] Brandão, F.G.S.L., Svore, K.M.: Quantum speed-ups for solving semidefinite programs. In: Proceedings of the 58th Symposium on Foundations of Computer Science (FOCS). pp. 415–426 (2017)

[26] Li, T., Chakrabarti, S., Wu, X.: Sublinear quantum algorithms for training linear and kernel-based classifiers. In: Proceedings of the 36th International Conference on Machine Learning (ICML). pp. 3815–3824 (2019)

[27] Clarkson, K., Hazan, E., Woodruff, D.: Sublinear optimization for machine learning. Journal of the ACM 59(5), 23 (2012)

[28] Apeldoorn, J.v., Gilyén, A.: Quantum algorithms for zero-sum games. Tech. Rep. arXiv: 1904.03180 (2019)

[29] Grigoriadis, M., Khachiyan, L.: A sublinear-time randomized approximation algorithm for matrix games. Operations Research Letters 18(2), 53–58 (1995)

[30] Jordan, S.P.: Fast quantum algorithm for numerical gradient estimation. Physical Review Letters 95, 050501 (2005)

[31] Gilyén, A., Arunachalam, S., Wiebe, N.: Optimizing quantum optimization algorithms via faster quantum gradient computation. In: Proceedings of the 30th Symposium on Discrete Algorithms (SODA). pp. 1425–1444 (2019)

[32] Apeldoorn, J.v., Gilyén, A., Gribling, S., de Wolf, R.: Convex optimization using quantum oracles. Tech. Rep. arXiv: 1809.00643 (2018)

[33] Chakrabarti, S., Childs, A.M., Li, T., Wu, X.: Quantum algorithms and lower bounds for convex optimization. Tech. Rep. arXiv: 1809.01731 (2018)

[34] Hazan, E., Kale, S.: Online submodular minimization. Journal of Machine Learning Research 13(1), 2903–2922 (2012)

[35] Vose, M.D.: A linear algorithm for generating random numbers with a given distribution. IEEE Transactions on Software Engineering 17(9), 972–975 (1991)

[36] Axelrod, B., Liu, Y.P., Sidford, A.: Near-optimal approximate discrete and continuous submodular function minimization. Tech. Rep. arXiv: 1909.00171 (2019)

[37] Harvey, N.J.A.: Matchings, Matroids and Submodular Functions. Ph.D. thesis, Cambridge, MA, USA (2008)

[38] Jegelka, S., Bilmes, J.A.: Online submodular minimization for combinatorial structures. In: Proceedings of the 28th International Conference on Machine Learning (ICML). pp. 345–352 (2011)

[39] Devroye, L.: Non-Uniform Random Variate Generation. Springer-Verlag (1986)

[40] Bratley, P., Fox, B.L., Schrage, L.E.: A Guide to Simulation. Springer-Verlag, second edn. (1987)

[41] Walker, A.J.: New fast method for generating discrete random numbers with arbitrary frequency distributions. Electronics Letters 10(8), 127–128 (1974)

[42] Grover, L.K.: Synthesis of quantum superpositions by quantum computation. Physical Review Letters 85, 1334–1337 (2000)

[43] Dürr, C., Høyer, P.: A quantum algorithm for finding the minimum. Tech. Rep. arXiv: 9607014 (1996)

[44] Brassard, G., Høyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Quantum Computation and Quantum Information: A Millennium Volume 1, 53–74 (2002)

[45] Sanders, Y.R., Low, G.H., Scherer, A., Berry, D.W.: Black-box quantum state preparation without arithmetic. Physical Review Letters 122, 020502 (2019)

[46] Brassard, G., Høyer, P., Mosca, M., Tapp, A.: Tight bounds on quantum searching. Fortschritte der Physik 46, 493–505 (1998)

[47] Duchi, J.C.: Introductory lectures on stochastic optimization. In: The Mathematics of Data, IAS/Park City Mathematics Series, vol. 25, pp. 99–185. American Mathematical Society (2018)

[48] Guibas, L.J., Sedgewick, R.: A dichromatic framework for balanced trees. In: Proceedings of the 19th Symposium on Foundations of Computer Science (FOCS). pp. 8–21 (1978)

Appendix A

6 Counterexample to the Gradient Sampling in [2]

Let $n = 2$, let $F : 2^{[2]} \to [-1, 1]$ be a submodular function and let $f : [0, 1]^2 \to \mathbb{R}$ be its Lovász extension. By definition (see Section 2), the Lovász subgradient of f at $x = (x_1, x_2) \in [0, 1]^2$ is

$$g(x_1, x_2) = \begin{cases} \big(F(\{1\}) - F(\varnothing),\, F(\{1,2\}) - F(\{1\})\big) & \text{if } x_1 \geq x_2 \\ \big(F(\{1,2\}) - F(\{2\}),\, F(\{2\}) - F(\varnothing)\big) & \text{if } x_1 < x_2. \end{cases}$$

Now let us consider a particular submodular function F. Let $F(\varnothing) = 0$, $F(\{1\}) = -\frac{1}{2}$, $F(\{2\}) = 0$, $F(\{1, 2\}) = -1$, and therefore

$$g(x_1, x_2) = \begin{cases} (-\frac{1}{2},\, -\frac{1}{2}) & \text{if } x_1 \geq x_2 \\ (-1,\, 0) & \text{if } x_1 < x_2. \end{cases}$$

The stochastic subgradient descent starts at $x^{(0)} = (0, 0)$, for which we have $g(x^{(0)}) = (-\frac{1}{2}, -\frac{1}{2})$ and $\|g(x^{(0)})\|_1 = 1$. Hence, with probability 1/2 each, we either have $\widetilde{g}^{(0)} = (-1, 0)$ or $\widetilde{g}^{(0)} = (0, -1)$. Since $x^{(1)} := \mathrm{argmin}_{x\in[0,1]^n} \|x - (x^{(0)} - \eta\widetilde{g}^{(0)})\|_2$ (for some step size parameter $0 < \eta < 1$), it follows that $x^{(1)} = (\eta, 0)$ or $x^{(1)} = (0, \eta)$, respectively. Suppose the latter is the case: $\widetilde{g}^{(0)} = (0, -1)$ and $x^{(1)} = (0, \eta)$. Then, the random variable $\widetilde{d}^{(1)}$ is an estimate of the subgradient difference $d^{(1)} = g(x^{(1)}) - g(x^{(0)}) = (-\frac{1}{2}, \frac{1}{2})$, namely, $\mathbb{E}[\widetilde{d}^{(1)} \mid x^{(1)}] = d^{(1)}$. Now, observe that the random variable $\widetilde{g}^{(1)} = \widetilde{g}^{(0)} + \widetilde{d}^{(1)}$ satisfies

$$\mathbb{E}[\widetilde{g}^{(1)} \mid x^{(1)}] = \mathbb{E}[\widetilde{g}^{(0)} \mid x^{(1)}] + \mathbb{E}[\widetilde{d}^{(1)} \mid x^{(1)}] = \widetilde{g}^{(0)} + d^{(1)} = (-1/2,\, -1/2) \neq (-1, 0) = g(x^{(1)}).$$

Hence, the procedure in [2] that returns $\widetilde{g}^{(t)}$ is not a valid subgradient oracle. We note that $x^{(1)} = (\eta, 0)$ would have led to the same error. This problem can be fixed by defining $\widetilde{g}^{(1)} = \widetilde{\widetilde{g}}^{(0)} + \widetilde{d}^{(1)}$ where $\widetilde{\widetilde{g}}^{(0)}$ is a second estimate of $g(x^{(0)})$ *independent* of $\widetilde{g}^{(0)}$, for instance $\widetilde{\widetilde{g}}^{(0)} = (-1, 0)$ or $\widetilde{\widetilde{g}}^{(0)} = (0, -1)$ with probability 1/2 each. In this case, we have $\mathbb{E}[\widetilde{g}^{(1)} \mid x^{(1)}] = g^{(0)} + d^{(1)} = g(x^{(1)})$.