Discrete Optimization 8 (2011) 459–477

Contents lists available at ScienceDirect

Discrete Optimization

journal homepage: www.elsevier.com/locate/disopt

On the complexity of submodular function minimisation on diamonds

Fredrik Kuivinen

Department of Computer and Information Science, Linköpings Universitet, SE-581 83, Linköping, Sweden

A R T I C L E I N F O

Article history:
Received 7 August 2009
Received in revised form 3 April 2011
Accepted 8 April 2011
Available online 14 May 2011

Keywords:
Combinatorial optimization
Submodular function minimization
Lattices
Diamonds

A B S T R A C T

Let $(L; \sqcap, \sqcup)$ be a finite lattice and let n be a positive integer. A function $f : L^n \rightarrow \mathbb{R}$ is said to be *submodular* if $f(\boldsymbol{a} \sqcap \boldsymbol{b}) + f(\boldsymbol{a} \sqcup \boldsymbol{b}) \leq f(\boldsymbol{a}) + f(\boldsymbol{b})$ for all $\boldsymbol{a}, \boldsymbol{b} \in L^n$. In this article we study submodular functions when L is a *diamond*. Given oracle access to f we are interested in finding $\boldsymbol{x} \in L^n$ such that $f(\boldsymbol{x}) = \min_{\boldsymbol{y} \in L^n} f(\boldsymbol{y})$ as efficiently as possible. We establish

- a min–max theorem, which states that the minimum of the submodular function is equal to the maximum of a certain function defined over a certain polyhedron; and
- a good characterisation of the minimisation problem, i.e., we show that given an oracle for computing a submodular $f : L^n \rightarrow \mathbb{Z}$ and an integer m such that $\min_{\boldsymbol{x} \in L^n} f(\boldsymbol{x}) = m$, there is a proof of this fact which can be verified in time polynomial in n and $\max_{\boldsymbol{t} \in L^n} \log |f(\boldsymbol{t})|$; and
- a pseudopolynomial-time algorithm for the minimisation problem, i.e., given an oracle for computing a submodular $f : L^n \rightarrow \mathbb{Z}$ one can find $\min_{\boldsymbol{t} \in L^n} f(\boldsymbol{t})$ in time bounded by a polynomial in n and $\max_{\boldsymbol{t} \in L^n} |f(\boldsymbol{t})|$.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Let V be a finite set and let f be a function from 2^V to $\mathbb{R}$. The function f is said to be *submodular* if $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$ for all $A, B \subseteq V$. In the sequel we will call such functions *submodular set functions*. Submodular set functions show up in various fields including combinatorial optimisation, graph theory [1], game theory [2], information theory [3] and statistical physics [4]. Examples include the cut function of graphs and the rank function of matroids. There is also a connection between submodular function minimisation and convex optimisation. In particular, submodularity can be seen as a discrete analogue of convexity [5,6]. We refer the reader to [7–9] for a general background on submodular set functions.

Given a submodular set function $f : 2^V \rightarrow \mathbb{R}$ there are several algorithms for finding minimisers of f, i.e., finding a subset $X \subseteq V$ such that $f(X) = \min_{Y \subseteq V} f(Y)$, in time polynomial in $|V|$. The first algorithm for finding such minimisers in polynomial time is due to Grötschel et al. [10]. However, this algorithm is based on the Ellipsoid algorithm and hence its usefulness in practice is limited. Almost two decades later two combinatorial algorithms were found independently by Schrijver [11] and Iwata et al. [12]. More recently the running times have been improved. The currently fastest strongly polynomial time algorithm is due to Orlin [13] and the fastest weakly polynomial time algorithm is due to Iwata [14]. In these algorithms the submodular set function is given by a value-giving oracle for f (i.e., presented with a subset $X \subseteq V$ the oracle computes $f(X)$).

In this article we investigate a more general notion of submodularity. Recall that a *lattice* is a partially ordered set in which each pair of elements has a least upper bound (join, $\sqcup$) and a greatest lower bound (meet, $\sqcap$). Given a finite lattice $\mathcal{L}$ (all lattices in this article are finite) and a positive integer n we can construct the *product lattice* $\mathcal{L}^n$. Meet and join for $\mathcal{L}^n$ are then defined coordinate-wise by meet and join in $\mathcal{L}$. We say that a function $h : \mathcal{L}^n \rightarrow \mathbb{R}$ is submodular if $h(\boldsymbol{a} \sqcap \boldsymbol{b}) + h(\boldsymbol{a} \sqcup \boldsymbol{b}) \leq h(\boldsymbol{a}) + h(\boldsymbol{b})$ for all $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{L}^n$. Note that the subsets of V can be seen as a lattice with union

E-mail addresses: frekui@gmail.com, freku@ida.liu.se.

1572-5286/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.disopt.2011.04.001

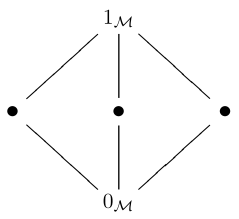

Fig. 1. The five element diamond.

as join and intersection as meet (this lattice is a product of the two element lattice). Hence, this notion of submodularity is a generalisation of submodular set functions. For a fixed finite lattice $\mathcal{L}$ we are interested in the submodular function minimisation (SFM) problem:

INSTANCE: An integer $n \geq 1$ and a submodular function f on $\mathcal{L}^n$.

GOAL: Find $\boldsymbol{x} \in \mathcal{L}^n$ such that $f(\boldsymbol{x}) = \min_{\boldsymbol{y} \in \mathcal{L}^n} f(\boldsymbol{y})$.

Following [15] we denote this problem by SFM($\mathcal{L}$). SFM($\mathcal{L}$) is said to be *oracle-tractable* if the problem can be solved in time polynomial in n (provided that we have access to a value-giving oracle for f and that we can assume that f is submodular, i.e., it is a promise problem). This definition naturally leads to the following question: is SFM($\mathcal{L}$) oracle-tractable for all finite lattices $\mathcal{L}$? (This question was, as far as we know, first asked by Cohen et al. [16].)

Schrijver [11] showed that given a sublattice S of 2^V (i.e., $S \subseteq 2^V$ and for any $X, Y \in S$ we have $X \cap Y, X \cup Y \in S$) and submodular function $f : S \to \mathbb{R}$ a minimiser of f can be found in time polynomial in n. In particular, this implies that for any distributive lattice $\mathcal{L}$ the problem SFM($\mathcal{L}$) is oracle-tractable. Krokhin and Larose [15] showed that certain constructions on lattices preserve oracle-tractability of SFM. In particular, they showed that if X is a class of lattices such that SFM($\mathcal{L}$) is oracle-tractable for every $\mathcal{L} \in X$, then so is SFM($\mathcal{L}'$) where $\mathcal{L}'$ is a *homomorphic image* of some lattice in X, a *direct product* of some lattices in X, or contained in the *Mal'tsev product* $X \circ X$. We will not define these constructions here and refer the reader to [15] instead.

A lattice $\mathcal{L}$ is a *diamond* if the elements of the lattice form a disjoint union of the bottom element $0_{\mathcal{L}}$, the top element $1_{\mathcal{L}}$, and a finite set A, $|A| \geq 3$ whose elements are pairwise incomparable. The elements in A are the *atoms* of the diamond. See Fig. 1 for a diagram of the five element diamond. We want to emphasise that diamonds have a different structure compared to the lattices defined by union and intersection. In particular, diamonds are not *distributive*, that is they do *not* satisfy $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$ for all $x, y, z \in \mathcal{L}$. We will denote the diamond with k atoms by $\mathcal{M}_k$. In Sections 4, 6 and 7 the complexity of SFM($\mathcal{M}_k$) is investigated. The reason for investigating the complexity of SFM on diamonds and not some other lattice, is that diamonds are (arguably) the simplest lattice that we do not have any complexity results for. In the approach taken in this article the difficult case is $k = 3$—the proofs for the $k = 3$ case generalise straightforwardly to an arbitrary k. We note that none of the diamonds are captured by the combination of the results found in [15,11] (a proof of this fact can be found in [15]).

Results and techniques. The first main result in this article is a min–max theorem for SFM($\mathcal{M}_k$) which is stated as Theorem 4.3. This result looks quite similar to Edmonds' min–max theorem for submodular set functions [17] (we present Edmonds' result in Section 2). The key step in the proof of this result is the definition of a certain polyhedron, which depends on f.

The second main result is a *good characterisation* of SFM($\mathcal{M}_k$) (Theorem 6.6). That is, we prove that given a submodular $f : \mathcal{M}_k^n \to \mathbb{Z}$ and integer m such that $\min_{\boldsymbol{x} \in \mathcal{L}^n} f(\boldsymbol{x}) = m$, there is a proof of this fact which can be verified in time polynomial in n and $\max_{\boldsymbol{y} \in \mathcal{L}^n} \log |f(\boldsymbol{y})|$ (under the assumption that f is submodular). This can be seen as placing SFM($\mathcal{M}_k$) in the appropriately modified variant of **NP** $\cap$ **coNP** (the differences from our setting to an ordinary optimisation problem is that we are given oracle access to the function to be minimised and we assume that the given function is submodular). The proof of this result makes use of Carathéodory's theorem and of the known polynomial-time algorithms for minimising submodular set functions. We also need our min–max theorem.

The third result is a pseudopolynomial-time algorithm for SFM($\mathcal{M}_k$) (see Section 7). We show that SFM($\mathcal{M}_k$) can be solved in time polynomial in n and $\max_{\boldsymbol{t} \in \mathcal{M}_k^n} |f(\boldsymbol{t})|$. The main part of the algorithm consists of a nested application of the Ellipsoid algorithm. We also need to prove that the polyhedrons we associate with submodular functions are half-integral. An interesting and challenging open problem is to construct an algorithm with a running time polynomial in n and $\max_{\boldsymbol{t} \in \mathcal{M}_k^n} \log |f(\boldsymbol{t})|$.

Our results apply to diamonds, however, as mentioned above, in [15] two constructions on lattices (Mal'tsev products and homomorphic images) are shown to preserve tractability results for SFM. By combining these constructions with the results in this article one gets tractability results for a much larger class of lattices than just diamonds.[1] In particular, by the results in this article there is a pseudopolynomial-time algorithm for minimising submodular functions over products of the lattice in Fig. 2.

[1] In [15] these constructions are shown to preserve oracle-tractability and not solvability in pseudopolynomial time. However, it is straightforward to adapt the proofs to the pseudopolynomial case.

Download English Version:

https://daneshyari.com/en/article/1141650

Download Persian Version:

https://daneshyari.com/article/1141650

Daneshyari.com