Can we measure the difficulty of an optimization problem?

Tansu Alpcan*, Tom Everitt**, and Marcus Hutter***

* Dept. of Electrical and Electronic Engineering
The University of Melbourne
** Department of Mathematics
Stockholm University
*** Research School of Computer Science
Australian National University

ITW 2014, Hobart

Outline

Introduction

- Can we measure the difficulty of an optimization problem?
- Although optimization plays a crucial role in modern science and technology, a formal framework that puts problems and solution algorithms into a broader context has not been established.
- This paper presents a **conceptual approach** which gives a positive answer to the question for a broad class of optimization problems.
- The proposed framework builds upon **Shannon and algorithmic information theories** and provides a computational perspective.

Overview

- A **concrete model** and definition **of a class of optimization problems** is provided.
- A **formal definition of optimization difficulty** is introduced which builds upon algorithmic information theory.
- Following an initial analysis, **lower and upper bounds** on optimization difficulty are established.
- One of the upper-bounds is closely related to Shannon information theory and black-box optimization.
- Finally, various **computational issues and future research** directions are discussed.

Optimization Problem

We consider mathematical optimization problems of type:

$$\max_x f(x) \text{ subject to } g_i(x) \leq 0,\ i = 1, \ldots, m,\ x \in \mathbb{R}^n.$$

- The list of constraints, denoted by c, define the solution space $\mathcal{A}$ which is a assumed to be a *compact* subset of $\mathcal{A} \subset \mathbb{R}^n$.
- Assume f is *Lipschitz-continuous* on $\mathcal{A}$ and there is a feasible *global solution* $x^* = \arg\max_{x \in \mathcal{A}} f(x)$.

Discretization

For a given scalar $\varepsilon > 0$ and compact set $\mathcal{A}$, let $\mathcal{A}(\varepsilon)$ be an *ε-discretization of $\mathcal{A}$* constructed using the following procedure:

1. Let $\mathcal{C}$ be a finite covering of $\mathcal{A}$ with hypercubes of side length at most ε.
2. For each cube $C \in \mathcal{C}$, let $x_C \in C \cap \mathcal{A}$.
3. Finally, let $\mathcal{A}(\varepsilon)$ be the set of all x_C, $C \in \mathcal{C}$.
4. Thus, $\mathcal{A}(\varepsilon)$ is a finite subset of $\mathcal{A}$, with the same cardinality as $\mathcal{C}$.
5. If the cubes in $\mathcal{C}$ do not overlap, we call discretization based on $\mathcal{C}$ *non-overlapping*.

Discrete Optimization Problem

Definition:
A (discretizable) optimization problem on $\mathbb{R}^n$ is a tuple $\langle f, c, \varepsilon \rangle$ where $f : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*, c is a list of constraints expressed using functional (in)equalities, and $\varepsilon > 0$ is a *discretization parameter*.

- Assume that the constraint set $\mathcal{A}$ is non-empty and compact and that f is Lipschitz-continuous over $\mathcal{A}$. Let $\mathcal{A}(\varepsilon)$ be an ε-discretization of $\mathcal{A}$.
- An *argmax* of f on $\mathcal{A}$ is a point $x^* \in \mathcal{A}$ satisfying $\forall x \in \mathcal{A} : f(x) \leq f(x^*)$.
- A *discrete argmax* (or δ_ε- argmax) is a point $\hat{x}$ satisfying $\forall x \in A : f(x) \leq f(\hat{x}) + \delta_\varepsilon$ where $\delta_\varepsilon = \max\{|f(x) - f(\hat{x})| : \|x - \hat{x}\| < \varepsilon\}$ and $\|\cdot\|$ is the maximum norm.

Discretized and Approximate Solutions

- The definition above accepts δ_ε- argmax (rather than true argmax) as a solution.
- The discretization parameter ε then effectively states how close the discrete argmax needs to be to the true argmax.
- If the Lipschitz constant of the objective function is k, then the desired solution differs at most δ in target value from the optimum, if one chooses $\varepsilon = \delta/k$.
- A sufficient condition for a $\hat{x} \in \mathcal{A}(\varepsilon)$ being a discrete argmax is: If it holds for all $x \in \mathcal{A}(\varepsilon)$ that $f(x) \leq f(\hat{x})$, then $\hat{x}$ must be a δ_ε- argmax.
- It is worth noting that *all numerical optimization software packages yield only discrete and approximate solutions*.

Binary Representation

- To allow for a computational treatment, an encoding of problems as (binary) strings must be chosen.
- The "standard calculus symbols" we base these descriptions on are:
  - finite precision real numbers $0, 1.354, \ldots;$
  - variables $x_1, x_2, \ldots;$
  - elementary functions $+, \cdot, \exp, \ldots;$
  - parenthesis;
  - relations $\leq, =, \ldots.$
- A *function* $\mathbb{R}^n \to \mathbb{R}$ is an expression formed by elementary functions, real numbers and the variables $x_1, \ldots, x_n$, and a *constraint on* $\mathbb{R}^n$ is a formula of the form $g(x_1, \ldots, x_n) \leq 0$ with $g : \mathbb{R}^n \to \mathbb{R}$.
- If e is an expression, let $\ell(e)$ denote the length of its binary encoding which can be obtained by giving each symbol a binary encoding (e.g. ASCII).

Correctness of Solutions

- To ensure correctness, any alleged δ_ε- argmax solution is paired with a polynomially verifiable certificate *s* of the the correctness.
- In general, the trace (step-by-step reporting) of a correct optimization algorithm forms one example of a (linearly verifiable) certificate.
- To verify, it suffices to check that each step of the trace corresponds to the definition of the algorithm, and that the final step of the trace outputs the proposed argmax.
- A general type of certificate (not specific to a particular class or optimization algorithm) may for example be based on formal proofs in first-order logic or type-theory. Many automated theorem proving systems have developed formalizations of analysis, which could potentially form the basis of a suitable proof system.

Verifiable Solutions

Definition:
Consider the discrete optimization problem and a suitable proof system $\mathcal{T}$ offering polynomially verifiable certificates s of candidate solutions. A *solution of the optimization problem* $\langle f, c, \varepsilon \rangle$ is defined as a pair $\langle x^*, s \rangle$ where s is a certificate in $\mathcal{T}$ that x^* is a δ_ε- argmax for $\langle f, c, \varepsilon \rangle$.

- It is beyond the scope of this paper to describe a suitable proof system $\mathcal{T}$ in detail. We will instead rely on semi-formal proof sketches in examples, and polynomial verifiability in abstract arguments.
- For concreteness, we will assume that certificates in $\mathcal{T}$ can be verified in time dn^q. That is, we assume the existence of a verifier for certificates in $\mathcal{T}$ with runtime at most dn^q for certificates of length n.

Illustrative Example

Consider the optimization problem

$$\langle f(x) = 3x^2 + x; x \geq -1, x \leq 1; \varepsilon = 0.001 \rangle . \qquad (1)$$

A solution is δ_ε- argmax $= 1$. The following informal certificate sketch validates the solution.

1. $df/dx = 6x + 1$ (derivative)
2. $6x + 1 = 0 \iff x = -1/6$ (properties of real numbers)
3. roots$(df/dx) = \{-1/6\}$ (from 1 and 2)
4. boundary $= \{-1, 1\}$ (from c)
5. $x \notin$ roots$(df/dx) \wedge x \notin$ boundary$(c) \implies \neg$ argmax(x) (calculus)
6. argmax $= -1/6 \vee$ argmax $= -1 \vee$ argmax $= 1$ (from 3–5)
7. $f(-1) \leq f(1) \implies$ argmax $\neq -1$
8. $f(-1/6) \leq f(1) \implies$ argmax $\neq -1/6$
9. argmax $= 1$ (from 6–8)
10. δ_ε- argmax $= 1$ (from 9)

Optimization Difficulty: Solution Concept

- For measuring its difficulty, the optimization problem is formulated as a “knowledge or “information” problem.
- Before solving $\langle f, c, \varepsilon \rangle$ it is only known that the solution has to be in the search domain, $\mathcal{A}$.
- Solving the optimization problem yields *knowledge* about the location of x^* up to a certain precision.
- Solving an optimization problem is equivalent to obtaining knowledge about the location of the solution.
- If a problem $\langle f, c, \varepsilon \rangle$ is “simple” then solving it corresponds to discovering a small amount of knowledge. Likewise, a difficult problem means a lot of knowledge is produced in solving it.

Optimization Algorithm

Definition:
An algorithm p *solves the optimization problem* $\langle f, c, \varepsilon \rangle$ if $p(\langle f, c, \varepsilon \rangle) = \langle x^*, s \rangle$ with s a certificate that x^* is a δ_ε- argmax of f on $\mathcal{A}$. That is, p should output a solution $\langle x^*, s \rangle$ when fed $\langle f, c, \varepsilon \rangle$ as input.

With U a universal Turing-machine (aka programming language), let the *description length* $\ell_U(p)$ be the length of the binary string-encoding of p on U, and let the *runtime* $t_U(p(\langle f, c, \varepsilon \rangle))$ be the number of time steps it takes for p to halt on input $\langle f, c, \varepsilon \rangle$ on U.

Optimization Difficulty

Definition:
The *optimization difficulty* of a given optimization problem $\langle f, c, \varepsilon \rangle$ is defined as

$$D_{\mathrm{opt}}(\langle f, c, \varepsilon \rangle) := \min_p \{\ell(p) + \log_2(t(p)) : p \text{ solves } \langle f, c, \varepsilon \rangle\}$$

Discussion:

- Definition above refers to *instances* of optimization problems rather than classes; multiple reasons for this choice.
- Generally applicable vs special solvers: certificates and "runtime-knowledge" vs "source code"-knowledge.

Optimization vs Search: Upper Bound 1

- A discretized optimization problem with a finite search space $\mathcal{A}(\varepsilon)$ can always be solved as a "search problem" by ignoring the properties of the objective function f.
- Assuming no a priori knowledge (i.e., a uniform prior over argmax-locations in $\mathcal{A}(\varepsilon)$), once the solution is found, the amount of a posteriori knowledge obtained is $\log_2(|\mathcal{A}(\varepsilon)|)$ bits from Shannon information theory.

Proposition III.3 (Upper Bound 1). There is a computational constant $k \in \mathbb{N}$ such that for any optimization problem $\langle f, c, \varepsilon \rangle$ with ε-discretization $\mathcal{A}(\varepsilon)$,

$$D_{\mathrm{opt}}(\langle f, c, \varepsilon \rangle) \leq k + \log_2(|\mathcal{A}(\varepsilon)| + C) \, ,$$

where C is the runtime cost of obtaining the discretization $\mathcal{A}(\varepsilon)$, and $|\cdot|$ denotes cardinality.

Optimization vs Search: Upper Bound 1

- The solution could also have been directly encoded into the source code of algorithm p. This way, p would not have had to perform the exhaustive search, reducing its runtime considerably.
- However, a standard result in algorithmic information theory is that the typical description length of an element $x \in \mathcal{A}(\varepsilon)$ is of order $\log_2(|\mathcal{A}(\varepsilon)|)$.
- Thus, the upper bound on D_{opt} would have been the same.
- This symmetry between information encoded in the algorithm, and information found searching, provides one deep justification of the particular combination "description-length plus the binary log of the runtime" used in the definition of D_{opt}.

Upper Bound 2

The difficulty of optimization is also bounded above by the shortest solution.

Proposition III.4 (Upper bound 2). There is a (small) computational constant k such that if $\langle f, c, \varepsilon \rangle$ is an optimization problem with shortest solution $\langle x^*, s \rangle$, then

$$D_{\mathrm{opt}}(\langle f, c, \varepsilon \rangle) \leq k + \ell(\langle x^*, s \rangle)$$

Proof: The proof is immediate: Let p be the program `Print '`$\langle x^*, s \rangle$`'.`
Note that the program in the proof is not as short as it initially looks. The program fits the entire solution s in the source code.

Lower Bound 1

The difficulty of optimization may also be bounded from below.

Proposition III.5 (Lower bound). Assume that $d \cdot n^q$ bounds the running time of the proof-verifier, and that $\langle f, c, \varepsilon \rangle$ is a non-trivial problem. Then

$$D_{\text{opt}}(\langle f, c, \varepsilon \rangle) \geq \frac{1}{q} \log_2(\ell(\langle f, c \rangle)) - \log_2(d)/q$$

The proof builds upon two bounds on the verification time of solutions for an optimal polynomial verifier v for the proof system $\mathcal{T}$.

Upper Bound 3

Proposition III.6 (Upper bound 3). There is a computational constant $k \in \mathbb{N}$ allowing the following bound: Let p output the correct argmax for all instances in a class S of optimization problems, and let s_p b a certificate for this. Let $\langle f, c, \varepsilon \rangle$ be a problem in S. Then

$$D_{\text{opt}}(\langle f, c, \varepsilon \rangle) \leq 2\ell(s_p) + \log_2(t(p(\langle f, c, \varepsilon \rangle))) + k + C$$

where C subsumes the cost of proving $\langle f, c, \varepsilon \rangle \in S$.
Note that, this connects the instance-difficulty introduced in this paper with the commonly-known class-difficulty from computational complexity theory, which is defined as the best (asymptotic) runtime of an algorithm outputting the correct argmax on all instances.

Conclusion

- The conceptual framework presented constitutes merely a first step in developing a deeper understanding of optimization problems from an information and algorithmic perspective.
- Several important issues are left for future analysis:
  - The intricate relationship between the description of the algorithm, its runtime, and the computing resources it requires.
  - Practical computability of optimization difficulty.
  - Availability of “information” about the optimization problem itself.
  - Approximations and noise.

Thank you and Questions?

Further information is available at:

http://www.tansu.alpcan.org
http://people.su.se/~toev8920