Theoretical Computer Science 567 (2015) 46–73

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

A framework for co-optimization algorithm performance and its application to worst-case optimization

Elena Popovici *, Ezra Winston

Icosystem Corp., 222 Third St., Suite 0142, Cambridge, MA, USA

ARTICLE INFO

Article history:
Received 5 September 2013
Received in revised form 2 October 2014
Accepted 21 October 2014
Available online 31 October 2014
Communicated by L. Kari

Keywords:
Co-optimization
Solution concepts
Performance
Optimality
Free lunch
Worst-case optimization
Best worst-case
Minimax
Maximin
Coevolution

ABSTRACT

Traditional black-box optimization searches a set of potential solutions for those optimizing the value of a function whose analytical or algebraic form is unknown or inexistent, but whose value can be queried for any input. Co-optimization is a generalization of this setting, in which fully evaluating a potential solution may require querying some function more than once, typically a very large number of times. When that's the case, co-optimization poses unique difficulties to designing and assessing algorithms. A generally-applicable approach is to judge co-optimization algorithm performance via an aggregate over all possible functions in the problem domain. We establish formal definitions of such aggregate performance and then investigate the following questions concerning algorithm design: 1) are some algorithms strictly better than others? i.e. is there "free lunch"? 2) do optimal algorithms exist? and 3) if so, are they practical? We formally define free lunch and aggregate optimality of co-optimization algorithms and derive generic conditions for their existence. We review and explain prior (no) free lunch results from the perspective of these conditions; we also show how this framework can be used to bridge several fields of research, by allowing formalization of their various problems and views on performance. We then apply and extend the generic results in a context involving a particular type of co-optimization called worst-case optimization. In this context we show that there exist algorithms that are aggregately-optimal for any budget (allowed number of function calls) and any starting point (set of previously uncovered function call outcomes), and also non-trivially strictly optimal for many budgets and starting points; moreover, we formalize the operation of such optimal algorithms and show that for certain domains, budgets and starting points this operation is equivalent to a simple procedure with tractable implementation—a first-of-its-kind result for co-optimization.

1. Introduction

In traditional optimization the goal is to find in a set of potential solutions the ones that optimize the value of a function; we say optimization is "black-box" if the function's analytical or algebraic form is unknown or inexistent, but the function's value can be queried for any input. Thus, determining whether one potential solution is better than another with respect to this goal involves only one function call per potential solution.

* Corresponding author.
E-mail addresses: elena@icosystem.com (E. Popovici), ezrawinston@gmail.com (E. Winston).

http://dx.doi.org/10.1016/j.tcs.2014.10.038

In co-optimization (also called generalized optimization [1]), the criterion specifying whether one potential solution is better than another, typically called a solution concept [2], may require a very large, possibly infinite number of function calls. For instance, this may be the case if potential solutions perform differently under different circumstances, and in practice it is often computationally prohibitive or even impossible to evaluate all function calls corresponding to even a single potential solution and every possible circumstance. As a result, an algorithm attempting to tackle such a co-optimization problem can only access incomplete information when choosing: a) which function calls to make—the exploration mechanism; and b) which potential solution looks best so far—the output mechanism. Since judging algorithms is typically based in some way on the potential solutions they output, assessing and comparing algorithm performance on a given function must also rely on incomplete information (apart from controlled research settings in which the function is a black-box only for the algorithm, but in fact analytically known to the researcher).

One way to make reliable performance comparisons under uncertainty is to aggregate over all possibilities for the unseen information [1]. Given such a notion of aggregate performance, the following questions are of immediate importance for designing high-performance co-optimization algorithms: 1) are some algorithms strictly better than others? 2) do optimal algorithms exist? and 3) if so, are they practical?

Most famously, the first question has been answered 'no' for certain traditional optimization contexts [3–8], a type of result called "no free lunch". By context we mean a specific combination of: solution concept, way of judging performance on a given function, method of aggregating over multiple functions, and any additional assumptions about the problem domain or the algorithms.

The first question has also been answered 'yes' for certain contexts, both in traditional optimization [8–12] and in non-traditional co-optimization [1,13–15]. Many of the latter results [1,13,14] involved worst-case optimization, i.e. the solution concept of finding the potential solution with best worst-case performance, which is of interest to game theory [16], function approximation [17,18], constraint optimization [17,19], robust discreet optimization [20], various kinds of engineering design and optimization [21,22] and scheduling [23–25].

When the answer to the first question is 'no', the answer to the second question is trivial: yes, all algorithms are aggregately-optimal. But if some algorithms are strictly better than others, meaning there *is* free lunch, then the second question is more interesting. For traditional optimization, it has been studied extensively. For non-traditional co-optimization, this question has been tackled only partially, via decomposition with respect to the two mechanisms forming the algorithm. Specifically, notions of optimal output mechanisms have been introduced for certain contexts [1,14,26,27]. However, these definitions were implicitly already based on assumptions of existence and tied to the specific context in which they were introduced in a way that didn't make it obvious which properties of that context guaranteed the existence. Additionally, most of these works focused on theoretically specifying what an aggregately-optimal output mechanism would have to output. Only in one case involving worst-case optimization has the theoretical definition been shown equivalent to a simple, easy-to-implement procedure [27]. To the best of our knowledge, this is to date the only result pertaining to the third question and there is no work defining or studying the existence and practicality of aggregately-optimal exploration mechanisms or complete algorithms for non-traditional co-optimization.

The goal of this paper is to fill some of these gaps in two-fold fashion: firstly, to establish a co-optimization performance framework unifying prior work, exposing open questions and facilitating future studies of additional contexts; secondly, to determine, via application of this framework, the existence, nature and tractability of aggregately-optimal algorithms—including exploration mechanisms—for worst-case optimization.

The rest of this manuscript is organized as follows. The next two sections pursue the first goal. In Section 2 we formalize co-optimization domains, problems, solution concepts and algorithms; we define worst-case optimization and show how it can be expressed as a co-optimization problem; we also show this for traditional single and multi-objective optimization and reinforcement learning, and make parallels to supervised machine learning. In Section 3 we formalize performance, free lunch and aggregate optimality and derive existence conditions; we link such notions for complete algorithms to the respective notions for the output and exploration mechanisms, and do so taking into account multiple possible budgets and starting points; we also exemplify how various views of performance can be instantiated from the framework and we explain, via the existence conditions, which context properties enabled prior results. Section 4 is a worst-case optimization study pursuing the second goal. Building upon the framework and results in Sections 2 and 3, we show a wide range of situations where it is easy to implement a worst-case co-optimization algorithm that is strictly aggregately-optimal for many budgets and starting points and this optimality is non-trivial, in that it consists of more than just avoiding duplicate function calls and exploiting lucky situations where the true worst-case performance of a potential solution is revealed. Section 5 provides discussion and conclusions. All proofs and tables summarizing notation are included in the accompanying supplementary materials.

2. Co-optimization framework

2.1. Domains and problems

Let us consider the real-world application described in [22] and summarized as follows. The design of large ships is concerned with building ships that are resilient to damages. Both the space of designs and the space of possible damages