Twenty-Two New Approximate Proof Labeling Schemes

Yuval Emek
Technion – Israel Institute of Technology, Haifa, Israel
yemek@technion.ac.il

Yuval Gil
Technion – Israel Institute of Technology, Haifa, Israel
yuval.gil@campus.technion.ac.il

Abstract

Introduced by Korman, Kutten, and Peleg (Distributed Computing 2005), a *proof labeling scheme (PLS)* is a system dedicated to verifying that a given configuration graph satisfies a certain property. It is composed of a centralized *prover*, whose role is to generate a proof for yes-instances in the form of an assignment of labels to the nodes, and a distributed *verifier*, whose role is to verify the validity of the proof by local means and accept it if and only if the property is satisfied. To overcome lower bounds on the label size of PLSs for certain graph properties, Censor-Hillel, Paz, and Perry (SIROCCO 2017) introduced the notion of an *approximate proof labeling scheme (APLS)* that allows the verifier to accept also some no-instances as long as they are not "too far" from satisfying the property.

The goal of the current paper is to advance our understanding of the power and limitations of APLSs. To this end, we formulate the notion of APLSs in terms of *distributed graph optimization problems (OptDGPs)* and develop two generic methods for the design of APLSs. These methods are then applied to various classic OptDGPs, obtaining twenty-two new APLSs. An appealing characteristic of our APLSs is that they are all *sequentially efficient* in the sense that both the prover and the verifier are required to run in (sequential) polynomial time. On the negative side, we establish "combinatorial" lower bounds on the label size for some of the aforementioned OptDGPs that demonstrate the optimality of our corresponding APLSs. For other OptDGPs, we establish conditional lower bounds that exploit the sequential efficiency of the verifier alone (under the assumption that NP $\neq$ co-NP) or that of both the verifier and the prover (under the assumption that P $\neq$ NP, with and without the unique games conjecture).

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of computation → Approximation algorithms analysis

Keywords and phrases proof labeling schemes, distributed graph problems, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.DISC.2020.20

Related Version A full version of the paper is available at `https://arxiv.org/abs/2007.14307` [10].

Funding This work has been supported by an Israeli Science Foundation grant number 1016/17.

1 Introduction

1.1 Model

Consider a connected undirected graph $G = (V, E)$ and denote $n = |V|$ and $m = |E|$. For a node $v \in V$, we stick to the convention that $N(v) = \{u \mid (u, v) \in E\}$ denotes the set of v's *neighbors* in G. An edge is said to be *incident* on v if it connects between v and one of its neighbors.

In the realm of *distributed graph algorithms*, the nodes of graph $G = (V, E)$ are associated with processing units that operate in a decentralized fashion. We assume that node $v \in V$ distinguishes between its incident edges by means of *port numbers*, i.e., a bijection between


© Yuval Emek and Yuval Gil;
licensed under Creative Commons License CC-BY

the set of edges incident on v and the integers in $\{1, ..., |N(v)|\}$. Additional graph attributes, such as node ids, edge orientation, and edge and node weights, are passed to the nodes by means of an *input assignment* $\mathsf{I} : V \to \{0,1\}^*$ that assigns to each node $v \in V$, a bit string $\mathsf{I}(v)$, referred to as v's *local input*, that encodes the additional attributes of v and its incident edges. The nodes return their output by means of an *output assignment* $\mathsf{O} : V \to \{0,1\}^*$ that assigns to each node $v \in V$, a bit string $\mathsf{O}(v)$, referred to as v's *local output*. We often denote $G_{\mathsf{I},\mathsf{O}} = \langle G, \mathsf{I}, \mathsf{O} \rangle$ and $G_{\mathsf{I}} = \langle G, \mathsf{I} \rangle$ and refer to these tuples as an *input-output (IO) graph* and an *input graph*, respectively.[1]

A *distributed graph problem (DGP)* Π is a collection of IO graphs $G_{\mathsf{I},\mathsf{O}}$. In the context of a DGP Π, an input graph G_{I} is said to be *legal* (and the graph G and input assignment I are said to be *co-legal*) if there exists an output assignment O such that $G_{\mathsf{I},\mathsf{O}} \in \Pi$, in which case we say that O is a *feasible solution* for G_{I} (or simply for G and I). Given a DGP Π, we may slightly abuse the notation and write $G_{\mathsf{I}} \in \Pi$ to denote that G_{I} is legal.

A *distributed graph minimization problem (MinDGP)* (resp., *distributed graph maximization problem (MaxDGP)*) Ψ is a pair $\langle \Pi, f \rangle$, where Π is a DGP and $f : \Pi \to \mathbb{Z}$ is a function, referred to as the *objective function* of Ψ, that maps each IO graph $G_{\mathsf{I},\mathsf{O}} \in \Pi$ to an integer value $f(G_{\mathsf{I},\mathsf{O}})$.[2] Given a co-legal graph G and input assignment I, define

$$OPT_{\Psi}(G, \mathsf{I}) = \inf_{\mathsf{O}: G_{\mathsf{I},\mathsf{O}} \in \Pi} \{f(G_{\mathsf{I},\mathsf{O}})\}$$

if Ψ is a MinDGP; and

$$OPT_{\Psi}(G, \mathsf{I}) = \sup_{\mathsf{O}: G_{\mathsf{I},\mathsf{O}} \in \Pi} \{f(G_{\mathsf{I},\mathsf{O}})\}$$

if Ψ is a MaxDGP. We often use the general term *distributed graph optimization problem (OptDGP)* to refer to MinDGPs as well as MaxDGPs. Given a OptDGP $\Psi = \langle \Pi, f \rangle$ and co-legal graph G and input assignment I, the output assignment O is said to be an *optimal solution* for G_{I} (or simply for G and I) if O is a feasible solution for G_{I} and $f(G_{\mathsf{I},\mathsf{O}}) = OPT_{\Psi}(G, \mathsf{I})$.

Let us demonstrate our definitions through the example of the maximum weight matching problem in bipartite graphs, i.e., explaining how it fits into the framework of a MaxDGP $\Psi = \langle \Pi, f \rangle$. Given a graph $G = (V, E)$ and an input assignment I, the input graph G_{I} is legal (with respect to Π) if G is bipartite and I encodes an edge weight function $w : E \to \mathbb{Z}$. Formally, for every node $v \in V$, the local input assignment $\mathsf{I}(v)$ is set to be a vector, indexed by the port numbers of v, defined so that if edge $e = (u, u') \in E$ corresponds to ports $1 \leq i \leq |N(u)|$ and $1 \leq i' \leq |N(u')|$ at nodes u and u', respectively, then both the i-th entry in $\mathsf{I}(u)$ and the i'-th entry in $\mathsf{I}(u')$ hold the value $w(e)$. Given a legal input graph $G_{\mathsf{I}} \in \Pi$, the output assignment O is a feasible solution for G_{I} if O encodes a matching $\mu \subseteq E$ in G. Formally, the local output assignment $\mathsf{O}(v)$ is set to the port number corresponding to e if there exists an edge $e \in \mu$ incident on v; and to $\perp$ otherwise. The objective function f of Ψ is defined so that for an IO graph $G_{\mathsf{I},\mathsf{O}} \in \Pi$ with corresponding edge weight function $w_{\mathsf{I}} : E \to \mathbb{Z}$ and matching $\mu_{\mathsf{O}} \subseteq E$, the value of f is set to $f(G_{\mathsf{I},\mathsf{O}}) = \sum_{e \in \mu_{\mathsf{O}}} w_{\mathsf{I}}(e)$. Following this notation, a feasible solution O for co-legal G and I is optimal if and only if μ_{O} is a maximum weight matching in G with respect to the edge weight function w_{I}.

While the formulation introduced in the current section is necessary for the general definitions presented in Section 1.1.1 and the generic methods developed in Section 3, in [10], when considering IO graphs in the context of specific DGPs and OptDGPs, we often do

[1] Refer to Table 1 for a full list of the abbreviations used in this paper.

[2] We assume for simplicity that the images of the objective functions used in the context of this paper, are integral. Lifting this assumption and allowing for real numerical values would complicate some of the arguments, but it does not affect the validity of our results.

Table 1 A list of abbreviations.

Term	Abbreviation	Reference
input-output graph	IO graph	Section 1.1
distributed graph problem	DGP	Section 1.1
distributed graph minimization problem	MinDGP	Section 1.1
distributed graph maximization problem	MaxDGP	Section 1.1
distributed graph optimization problem	OptDGP	Section 1.1
gap proof labeling scheme	GPLS	Section 1.1.1
proof labeling scheme	PLS	Section 1.1.1
approximate proof labeling scheme	APLS	Section 1.1.1
decision proof labeling scheme	DPLS	Section 1.1.1
approximate decision proof labeling scheme	ADPLS	Section 1.1.1
verifiable centralized approximation	VCA	Section 3.2

not explicitly describe the input and output assignments, but rather take a more natural high-level approach. For example, in the context of the aforementioned maximum weight matching problem in bipartite graphs, we may address the input edge weight function and output matching directly without providing an explanation as to how they are encoded in the input and output assignments, respectively. The missing details would be clear from the context and could be easily completed by the reader.

1.1.1 Proof Labeling Schemes

In this section we present the notions of proof labeling schemes [26] and approximate proof labeling schemes [5] for OptDGPs and their decision variants. To unify the definitions of these notions, we start by introducing the notion of gap proof labeling schemes based on the following definition.

A *configuration graph* $G_S = \langle G, S \rangle$ is a pair consisting of a graph $G = (V, E)$ and a function $S : V \to \{0,1\}^*$ assigning a bit string $S(v)$ to each node $v \in V$. In particular, an input graph G_{I} is a configuration graph, where $S(v) = \mathsf{I}(v)$, and an IO graph $G_{\mathsf{I},\mathsf{O}}$ is a configuration graph, where $S(v) = \mathsf{I}(v) \cdot \mathsf{O}(v)$.

Fix some universe $\mathcal{U}$ of configuration graphs. A *gap proof labeling scheme (GPLS)* is a mechanism designed to distinguish the configuration graphs in a *yes-family* $\mathcal{F}_Y \subset \mathcal{U}$ from the configuration graphs in a *no-family* $\mathcal{F}_N \subset \mathcal{U}$, where $\mathcal{F}_Y \cap \mathcal{F}_N = \emptyset$. This is done by means of a (centralized) *prover* and a (distributed) *verifier* that play the following roles: Given a configuration graph $G_S \in \mathcal{U}$, if $G_S \in \mathcal{F}_Y$, then the prover assigns a bit string $L(v)$, called the *label* of v, to each node $v \in V$. Let $L^N(v)$ be the vector of labels assigned to v's neighbors. The verifier at node $v \in V$ is provided with the 3-tuple $\langle S(v), L(v), L^N(v) \rangle$ and returns a Boolean value $\varphi(v)$.

We say that the verifier *accepts* G_S if $\varphi(v) = \texttt{True}$ for all nodes $v \in V$; and that the verifier *rejects* G_S if $\varphi(v) = \texttt{False}$ for at least one node $v \in V$. The GPLS is said to be *correct* if the following requirements hold for every configuration graph $G_S \in \mathcal{U}$:

▶ **R1.** *If $G_S \in \mathcal{F}_Y$, then the prover produces a label assignment $L : V \to \{0,1\}^*$ such that the verifier accepts G_S.*

▶ **R2.** *If $G_S \in \mathcal{F}_N$, then for any label assignment $L : V \to \{0,1\}^*$, the verifier rejects G_S.*

We emphasize that no requirements are made for configuration graphs $G_S \in \mathcal{U} \setminus (\mathcal{F}_Y \cup \mathcal{F}_N)$; in particular, the verifier may either accept or reject these configuration graphs (the same holds for configuration graphs that do not belong to the universe $\mathcal{U}$).

The performance of a GPLS is measured by means of its *proof size* defined to be the maximum length of a label $L(v)$ assigned by the prover to the nodes $v \in V$ assuming that $G_S \in \mathcal{F}_Y$. We say that GPLS admits a *sequentially efficient prover* if for any configuration graph $G_S \in F_Y$, the sequential runtime of the prover is polynomial in the number of bits used to encode G_S; and that it admits a *sequentially efficient verifier* if the sequential runtime of the verifier in node $v \in V$ is polynomial in $|S(v)|$, $|L(v)|$, and $\sum_{u \in N(v)} |L(u)|$. The GPLS is called *sequentially efficient* if both its prover and verifier are sequentially efficient.

Proof Labeling Schemes for OptDGPs. Consider some OptDGP $\Psi = \langle \Pi, f \rangle$ and let $\mathcal{U} = \{G_{\mathsf{I},\mathsf{O}} \mid G_{\mathsf{I}} \in \Pi\}$. A *proof labeling scheme (PLS)* for Ψ is defined as a GPLS over $\mathcal{U}$ by setting the yes-family to be

$$\mathcal{F}_Y = \{G_{\mathsf{I},\mathsf{O}} \in \Pi \mid f(G_{\mathsf{I},\mathsf{O}}) = OPT_\Psi(G, \mathsf{I})\}$$

and the no-family to be $\mathcal{F}_N = \mathcal{U} \setminus \mathcal{F}_Y$. In other words, a PLS for Ψ determines for a given IO graph $G_{\mathsf{I},\mathsf{O}} \in \mathcal{U}$ whether the output assignment $\mathsf{O} : V \to \{0,1\}^*$ is an optimal solution (which means in particular that it is a feasible solution) for the co-legal graph $G = (V, E)$ and input assignment $\mathsf{I} : V \to \{0,1\}^*$.

In the realm of OptDGPs, it is natural to relax the definition of a PLS so that it may also accept feasible solutions that only approximate the optimal ones. Specifically, given an approximation parameter $\alpha \geq 1$, an *α-approximate proof labeling scheme (α-APLS)* for a OptDGP $\Psi = \langle \Pi, f \rangle$ is defined in the same way as a PLS for Ψ with the sole difference that the no-family is defined by setting

$$\mathcal{F}_N = \begin{cases} \mathcal{U} \setminus \{G_{\mathsf{I},\mathsf{O}} \in \Pi \mid f(G_{\mathsf{I},\mathsf{O}}) \leq \alpha \cdot OPT_\Psi(G, \mathsf{I})\} , & \text{if } \Psi \text{ is a MinDGP} \\ \mathcal{U} \setminus \{G_{\mathsf{I},\mathsf{O}} \in \Pi \mid f(G_{\mathsf{I},\mathsf{O}}) \geq OPT_\Psi(G, \mathsf{I})/\alpha\} , & \text{if } \Psi \text{ is a MaxDGP} \end{cases} .$$

Decision Proof Labeling Schemes for OptDGPs. Consider some MinDGP (resp., MaxDGP) $\Psi = \langle \Pi, f \rangle$ and let $\mathcal{U} = \{G_{\mathsf{I}} \mid G_{\mathsf{I}} \in \Pi\}$. A *decision proof labeling scheme (DPLS)* for Ψ and a parameter $k \in \mathbb{Z}$ is defined as a GPLS over $\mathcal{U}$ by setting the yes-family to be

$$\mathcal{F}_Y = \begin{cases} \{G_{\mathsf{I}} \in \Pi \mid OPT_\Psi(G, \mathsf{I}) \geq k\} , & \text{if } \Psi \text{ is a MinDGP} \\ \{G_{\mathsf{I}} \in \Pi \mid OPT_\Psi(G, \mathsf{I}) \leq k\} , & \text{if } \Psi \text{ is a MaxDGP} \end{cases}$$

and the no-family to be $\mathcal{F}_N = \mathcal{U} \setminus \mathcal{F}_Y$. In other words, given an input graph $G_{\mathsf{I}} \in \Pi$, a DPLS for Ψ and k decides if $f(G_{\mathsf{I},\mathsf{O}}) \geq k$ (resp., $f(G_{\mathsf{I},\mathsf{O}}) \leq k$) for every feasible output assignment $\mathsf{O} : V \to \{0,1\}^*$. Notice that while PLSs address the task of verifying the optimality of a given output assignment O, that is, verifying that no output assignment admits an objective value smaller (resp., larger) than $f(G_{\mathsf{I},\mathsf{O}})$, in DPLSs, the output assignment O is not specified and the task is to verify that no output assignment admits an objective value smaller (resp., larger) than the parameter k, provided as part of the DPLS task.

Similarly to PLSs, the definition of DPLS admits a natural relaxation. Given an approximation parameter $\alpha \geq 1$, an *α-approximate decision proof labeling scheme (α-ADPLS)* for a OptDGP $\Psi = \langle \Pi, f \rangle$ and a parameter $k \in \mathbb{Z}$ is defined in the same way as a DPLS for Ψ and k with the sole difference that the no-family is defined by setting

$$\mathcal{F}_N = \begin{cases} \mathcal{U} \setminus \{G_{\mathsf{I}} \in \Pi \mid OPT_\Psi(G, \mathsf{I}) \geq k/\alpha\} , & \text{if } \Psi \text{ is a MinDGP} \\ \mathcal{U} \setminus \{G_{\mathsf{I}} \in \Pi \mid OPT_\Psi(G, \mathsf{I}) \leq \alpha \cdot k\} , & \text{if } \Psi \text{ is a MaxDGP} \end{cases} .$$

We often refer to an α-ADPLS without explicitly mentioning its associated parameter k; this should be interpreted with a universal quantifier over all parameters $k \in \mathbb{Z}$.

1.2 Related Work and Discussion

Distributed verification is the task of locally verifying a global property of a given configuration graph by means of a centralized prover and a distributed verifier. Various models for distributed verification have been introduced in the literature including the PLS model [26] as defined in Section 1.1.1, the *locally checkable proofs (LCP)* model [18], and the distributed complexity class *non-deterministic local decision (NLD)* [15, 3]. Refer to [12] for a comprehensive survey on the topic of distributed verification.

The current paper focuses on the PLS (and DPLS) model. This model was introduced by Korman, Kutten, and Peleg in [26] and has been extensively studied since then, see, e.g., [25, 4, 29, 13, 30, 14, 11]. A specific family of tasks that attracted a lot of attention in this regard is that of designing PLSs for classic optimization problems. Papers on this topic include [25], where a PLS for minimum spanning tree is shown to have a proof size of $O(\log n \log W)$, where W is the maximum weight, and [18], where a PLS for maximum weight matching in bipartite graphs is shown to have a proof size of $O(\log W)$.

In parallel, numerous researchers focused on establishing impossibility results for PLSs and DPLSs, usually derived from non-deterministic communication complexity lower bounds [27]. Such results are provided, e.g., in [2], where a proof size of $\tilde{\Omega}(n^2)$ is shown to be required for many classic optimization problems, and in [18], where an $\Omega(n^2/\log n)$ lower bound is established on the proof size of DPLSs for the problem of deciding if the chromatic number is larger than 3. For the minimum spanning tree problem, the authors of [25] proved that their $O(\log n \log W)$ upper bound on the proof size is asymptotically optimal, relying on direct combinatorial arguments.

The lower bounds on the proof size of PLSs (and DPLSs) for some optimization problems have motivated the authors of [5] to introduce the APLS (and ADPLS) notion as a natural relaxation thereof. This motivation is demonstrated by the task of verifying that the unweighted diameter of a given graph is at most k: As shown in [5], the diameter task admits a large gap between the required proof size of a DPLS, shown to be $\Omega(n/k)$, and the proof sizes of (3/2)-ADPLS and 2-ADPLS shown to be $O(\sqrt{n}\log^2 n)$ and $O(\log n)$, respectively. To the best of our knowledge, APLSs (and ADPLSs) have not been studied otherwise until the current paper.

One of the generic methods developed in the current paper for the design of APLSs for an abstract OptDGP relies on a *primal dual* approach applied to the linear program that encodes the OptDGP, after relaxing its integrality constraints (see Section 3.1). This can be viewed as a generalization of a similar approach used in the literature for concrete OptDGPs. Specifically, this primal dual approach is employed in [18] to obtain their PLS for maximum weight matching in bipartite graphs with a proof size of $O(\log W)$. A similar technique is used by the authors of [5] to achieve a 2-APLS for maximum weight matching in general graphs with the same proof size.

While most of the PLS literature (including the current work) focuses on deterministic schemes, an interesting angle that has been studied recently is randomization in distributed proofs, i.e., allowing the verifier to reach its decision in a randomized fashion. The notion of *randomized proof labeling schemes* was introduced in [17], where the strength of randomization in the PLS model is demonstrated by a universal scheme that enables one to reduce the amount of required communication in a PLS exponentially by allowing a (probabilistic) one-sided error. Another interesting generalization of PLSs is the *distributed interactive proof* model, introduced recently in [24] and studied further in [28, 7, 16].

On Sequential Efficiency. In this paper, we focus on sequentially efficient schemes, restricting the prover and verifier to "reasonable computations". We argue that beyond the interesting theoretical implications of this restriction (see Section 1.3), it also carries practical justifications: A natural application of PLSs is found in *local checking* for self-stabilizing algorithms [1], where the verifier's role is played by the detection module and the prover is part of the correction module [26]. Any attempt to implement these modules in practice clearly requires sequential efficiency on behalf of both the verifier and the prover (although, for the latter, the sequential efficiency condition alone is not sufficient as the correction module is also distributed).

While most of the PLSs presented in previous papers are naturally sequentially efficient, there are a few exceptions. One example of a scheme that may require intractable computations on the verifier side is the universal PLS presented in [26] that enables the verification of any decidable graph property with a label size of $O(n^2)$ simply by encoding the entire structure of the graph within the label. A PLS that inherently relies on sequentially inefficient prover can be found, e.g., in [18], where a scheme is constructed to decide if the graph contains a Hamiltonian cycle.

1.3 Our Contribution

Our goal in this paper is to explore the power and limitations of APLSs and ADPLS for OptDGPs. We start by developing two generic methods: a primal dual method for the design of sequentially efficient APLSs that expands and generalizes techniques used by Göös and Suomela [18] and Censor-Hillel, Paz, and Perry [5]; and a method that exploits the local properties of centralized approximation algorithms for the design of sequentially efficient ADPLSs. Next, we establish black-box reductions between APLSs and ADPLSs for certain families of OptDGPs. Based (mainly) on these generic methods and reductions, we design a total of twenty-two new sequentially efficient APLSs and ADPLSs for various classic optimization problems; refer to Tables 2 and 3 for a summary of these results.

On the negative side, we establish an $\Omega(\log \kappa)$ lower bound on the proof size of a $\frac{\kappa+1}{\kappa}$-APLS for maximum b-matching (in fact, this lower bound applies even for the simpler case of maximum matching) and minimum edge cover in graphs of odd-girth $2\kappa + 1$; and an $\Omega(\log n)$ lower bound on the proof size of a PLS for minimum edge cover in odd rings. These lower bounds, that rely on combinatorial arguments and hold regardless of sequential efficiency, match the proof size established in our corresponding APLSs for these OptDGPs, thus proving their optimality.

Additional lower bounds are established under the restriction of the verifier and/or prover to sequentially efficient computations, based on hardness assumptions in (sequential) computational complexity theory. Consider a OptDGP Ψ that corresponds to an optimization problem that is NP-hard to approximate within $\alpha \geq 1$. We first note that under the assumption that NP $\neq$ co-NP, the yes-families of both an α-APLS for Ψ and an α-ADPLS for Ψ (with some parameter $k \in \mathbb{Z}$) are languages in the complexity class co-NP $\setminus$ NP. Therefore, restricting the verifier to sequentially efficient computations implies that Ψ admits neither an α-APLS, nor an α-ADPLS, with a polynomial proof size. This provides additional motivation for the study of APLSs and ADPLSs over their exact counterparts.

Furthermore, the (weaker) assumption that P $\neq$ NP suffices to rule out the existence of α-ADPLS for Ψ when both the verifier and prover are required to be sequentially efficient. This is due to the fact that the yes-family of an α-ADPLS for Ψ (with some parameter $k \in \mathbb{Z}$) is a co-NP complete language, combined with the trivial observation that any sequentially efficient GPLS can be simulated by a centralized algorithm in polynomial time. We note

Table 2 α-APLS results and proof sizes. Refer to [10, Table 2] for references to these results.

Problem	Graph family	α	Proof size
minimum edge cover	odd-girth $\geq 2\kappa+1$	$(\kappa+1)/\kappa$	$O(\log \kappa)$
	odd-girth $\geq 2\kappa+1$	$(\kappa+1)/\kappa$	$\Omega(\log \kappa)$
	odd rings	1	$O(\log n)$
	odd rings	1	$\Omega(\log n)$
	bipartite	1	1
maximum b-matching	odd-girth $\geq 2\kappa+1$	$(\kappa+1)/\kappa$	$O(\log \kappa)$
	odd-girth $\geq 2\kappa+1$	$(\kappa+1)/\kappa$	$\Omega(\log \kappa)$
	bipartite	1	1
min-weight vertex cover	any	2	$O(\log n + \log W)$
min-weight dominating set	any	$\ln(n)+1$	$O(\log n + \log W)$
traveling salesperson	metric	2	$O(\log n + \log W)$
minimum Steiner tree	metric	2	$O(\log n + \log W)$
maximum flow	directed	1	1
max-weight cut	any	2	1

Table 3 α-ADPLS results and proof sizes. Refer to [10, Table 3] for references to these results.

Problem	Graph family	α	Proof size
minimum edge cover	odd-girth $\geq 2\kappa+1$	$(\kappa+1)/\kappa$	$O(\log n)$
	odd rings	1	$O(\log n)$
	bipartite	1	$O(\log n)$
maximum b-matching	odd-girth $\geq 2\kappa+1$	$(\kappa+1)/\kappa$	$O(\log n + \log W)$
	bipartite	1	$O(\log n + \log W)$
min-weight vertex cover	any	2	$O(\log n + \log W)$
min-weight dominating set	any	$\ln(n)+1$	$O(\log n + \log W)$
traveling salesperson	metric	2	$O(\log n + \log W)$
minimum Steiner tree	metric	2	$O(\log n + \log W)$
maximum flow	directed	1	$O(\log n + \log W)$
max-weight cut	any	2	$O(\log n + \log W)$

that most of the OptDGPs considered in this paper correspond to NP-hard optimization problems; refer to Table 4 for their known inapproximability results with and without the unique games conjecture [21].

1.4 Paper's Organization

The rest of the paper is organized as follows. Following some preliminaries presented in Section 2, our generic methods for the design of APLSs and ADPLSs are developed in Section 3. The reductions between APLSs and ADPLSs are presented in Section 4. Finally, the bounds we establish for concrete OptDGPS are established in [10].

2 Preliminaries

Linear Programming and Duality. A *linear program (LP)* consists of a linear objective function that one wishes to optimize (i.e., minimize or maximize) subject to linear inequality constraints. The *standard form* of a minimization (resp., maximization) LP is $\min\{\mathbf{c}^{\mathrm{T}}\mathbf{x} \mid$

Table 4 Known inapproximability bounds with and without the unique games conjecture.

Problem	Inapproximability	Inapproximability w. UGC
min-weight vertex cover	$1.36 - \varepsilon$ [8]	$2 - \varepsilon$ [23]
min-weight dominating set	$(1 - o(1)) \cdot \ln n$ [9]	
metric traveling salesperson	$123/122 - \varepsilon$ [20]	
metric minimum Steiner tree	$96/95 - \varepsilon$ [6]	
max-weight cut	$1.063 - \varepsilon$ [19]	$1.139 - \varepsilon$ [22]

$\mathbf{Ax} \geq \mathbf{b} \wedge \mathbf{x} \geq \mathbf{0}\}$ (resp., $\max\{\mathbf{c}^{\mathrm{T}}\mathbf{x} \mid \mathbf{Ax} \leq \mathbf{b} \wedge \mathbf{x} \geq \mathbf{0}\}$), where $\mathbf{x} = \{x_j\} \in \mathbb{R}^{\ell}$ is a vector of variables and $\mathbf{A} = \{a_{i,j}\} \in \mathbb{R}^{k \times \ell}$, $\mathbf{b} = \{b_i\} \in \mathbb{R}^k$, and $\mathbf{c} = \{c_j\} \in \mathbb{R}^{\ell}$ are a matrix and vectors of coefficients. An *integer linear program (ILP)* is a LP augmented with integrality constraints. In [10], we formulate OptDGPs as LPs and ILPs. In the latter case, we often turn to a *LP relaxation* of the problem, i.e., a LP obtained from an ILP by relaxing its integrality constraints.

Every LP admits a corresponding *dual program* (in this context, we refer to the original LP as the *primal program*). Specifically, for a minimization (resp., maximization) LP in standard form, its dual is a maximization (resp., minimization) LP, formulated as $\max\{\mathbf{b}^{\mathrm{T}}\mathbf{y} \mid \mathbf{A}^{\mathrm{T}}\mathbf{y} \leq \mathbf{c} \wedge \mathbf{y} \geq \mathbf{0}\}$ (resp., $\min\{\mathbf{b}^{\mathrm{T}}\mathbf{y} \mid \mathbf{A}^{\mathrm{T}}\mathbf{y} \geq \mathbf{c} \wedge \mathbf{y} \geq \mathbf{0}\}$).

LP duality has the following useful properties. Let $\mathbf{x}$ and $\mathbf{y}$ be feasible solutions to the primal and dual programs, respectively. The *weak duality* theorem states that $\mathbf{c}^{\mathrm{T}}\mathbf{x} \geq \mathbf{b}^{\mathrm{T}}\mathbf{y}$ (resp., $\mathbf{c}^{\mathrm{T}}\mathbf{x} \leq \mathbf{b}^{\mathrm{T}}\mathbf{y}$). The *strong duality* theorem states that $\mathbf{x}$ and $\mathbf{y}$ are optimal solutions to the primal and dual programs, respectively, if and only if $\mathbf{c}^{\mathrm{T}}\mathbf{x} = \mathbf{b}^{\mathrm{T}}\mathbf{y}$. The *relaxed complementary slackness* conditions are stated as follows, for given parameters $\beta, \gamma \geq 1$.

- Primal relaxed complementary slackness:

  For every primal variable x_j, if $x_j > 0$, then $c_j/\beta \leq \sum_{i=1}^{k} a_{ij} y_i \leq c_j$ (resp., $c_j \leq \sum_{i=1}^{k} a_{ij} y_i \leq \beta \cdot c_j$).
- Dual relaxed complementary slackness:

  For every dual variable y_i, if $y_i > 0$, then $b_i \leq \sum_{j=1}^{\ell} a_{ij} x_j \leq \gamma \cdot b_i$ (resp., $b_i/\gamma \leq \sum_{j=1}^{\ell} a_{ij} x_j \leq b_i$).

If the (primal and dual) relaxed complementary slackness conditions hold, then it is guaranteed that $\mathbf{c}^{\mathrm{T}}\mathbf{x} \leq \beta \cdot \gamma \cdot \mathbf{b}^{\mathrm{T}}\mathbf{y}$ (resp., $\mathbf{c}^{\mathrm{T}}\mathbf{x} \geq \frac{1}{\beta \cdot \gamma} \cdot \mathbf{b}^{\mathrm{T}}\mathbf{y}$) which, combined with the aforementioned weak duality theorem, implies that $\mathbf{x}$ approximates an optimal primal solution by a multiplicative factor of $\beta \cdot \gamma$. Moreover, the relaxed complementary slackness conditions with parameters $\beta = \gamma = 1$, often referred to simply as the *complementary slackness* conditions, hold if and only if $\mathbf{x}$ and $\mathbf{y}$ are optimal.

Let $\Psi = \langle \Pi, f \rangle$ be a OptDGP that can be represented as an ILP. Let P be its LP relaxation and D the dual LP of P. Given parameters $\beta, \gamma \geq 1$, we say that Ψ is *(β, γ)-fitted* if for any optimal (integral) solution $\mathbf{x}$ for the ILP corresponding to P, there exists a feasible solution $\mathbf{y}$ for D such that the relaxed primal and dual complementary slackness conditions hold for $\mathbf{x}$ and $\mathbf{y}$ with parameters β and γ, respectively.

Comparison Schemes. Let $\mathcal{U}$ be the universe of IO graphs $G_{\mathsf{I},\mathsf{O}}$ where $\mathsf{I} : V \to \{0,1\}^*$ is an input assignment that encodes a unique id represented using $O(\log n)$ bits for each node $v \in V$ (possibly among other input components). For a function $h : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ and parameter $k \in \mathbb{Z}$, an *(h,k)-comparison scheme* is a mechanism designed to decide if

$\sum_{v\in V} h(\mathsf{I}(v),\mathsf{O}(v)) \geq k$ for a given IO graph $G_{\mathsf{I},\mathsf{O}} \in \mathcal{U}$. Formally, an (h,k)-comparison scheme is defined as a GPLS over $\mathcal{U}$ by setting the yes-family to be $\mathcal{F}_Y = \{G_{\mathsf{I},\mathsf{O}} \in \mathcal{U} \mid \sum_{v\in V} h(\mathsf{I}(v),\mathsf{O}(v)) \geq k\}$ and the no family to be $\mathcal{F}_N = \mathcal{U} \setminus \mathcal{F}_Y$. Notice that the task of deciding if $\sum_{v\in V} h(\mathsf{I}(v),\mathsf{O}(v)) \leq k$ can be achieved by a $(-h,-k)$-comparison scheme, where $-h : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ is defined by setting $-h(a,b) = -1 \cdot h(a,b)$ for every $a,b \in \{0,1\}^*$.

The following lemma has been established by Korman et al. [26, Lemma 4.4].

▶ **Lemma 2.1.** *Given a function $h : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ that is computable in polynomial time and an integer $k \in \mathbb{Z}$, there exists a sequentially efficient (h,k)-comparison scheme with proof size $O(\log n + H)$, where H is the maximal number of bits required to represent $h(\mathsf{I}(v),\mathsf{O}(v))$ for any $v \in V$.*

Additional Definitions. A *feasibility scheme* for a DGP Π is a GPLS over the universe $\mathcal{U} = \{G_{\mathsf{I},\mathsf{O}} \mid G_{\mathsf{I}} \in \Pi\}$ with the yes-family $\mathcal{F}_Y = \Pi$ and the no-family $\mathcal{U} \setminus \mathcal{F}_Y$. The *odd-girth* of a graph $G = (V,E)$ is the length of the shortest odd cycle contained in G.

3 Methods

In this section, we present two generic methods that facilitate the design of sequentially efficient APLSs and ADPLSs with small proof sizes for many OptDGPs. These methods are used in most of the results established later on in [10].

3.1 The Primal Dual Method

LP duality theory can be a useful tool in the design of a $(\beta\cdot\gamma)$-APLS for a (β,γ)-fitted OptDGP Ψ (as shown in [5, 18]). The main idea of this approach is to use the relaxed complementary slackness conditions to verify that the output assignment $\mathsf{O} : V \to \{0,1\}^*$ of a given IO graph $G_{\mathsf{I},\mathsf{O}}$ is approximately optimal for G and I with respect to Ψ. Specifically, the prover provides the verifier with a proof that there exists a feasible dual solution $\mathbf{y}$ within a multiplicative factor of $\beta\cdot\gamma$ from the primal solution $\mathbf{x}$ derived from the output assignment O; the verifier then verifies the primal and dual feasibility of $\mathbf{x}$ and $\mathbf{y}$, respectively, as well as their relaxed complementary slackness conditions.

We take a particular interest in the following family of OptDGPs. Consider a OptDGP $\Psi = \langle \Pi, f\rangle$ that can be represented by an ILP that admits a LP relaxation P whose matrix form is given by the variable vector $\mathbf{x} = \{x_j\} \in \mathbb{R}^\ell$ and coefficient matrix and vectors $\mathbf{A} = \{a_{i,j}\} \in \mathbb{R}^{k\times\ell}$, $\mathbf{b} = \{b_i\} \in \mathbb{R}^k$, and $\mathbf{c} = \{c_j\} \in \mathbb{R}^\ell$. We say that Ψ is *locally verifiable* if for every IO graph $G_{\mathsf{I},\mathsf{O}} \in \Pi$, there exist mappings $v : [k] \to V$ and $e : [\ell] \to E$ that satisfy the following conditions: (1) $a_{i,j} = 0$ for every $i \in [k]$ and $j \in [\ell]$ such that $e(j)$ is not incident on $v(i)$; (2) the variable x_j is encoded in the local output $\mathsf{O}(u)$ of node $u \in V$ for every $j \in [k]$ such that $e(j)$ is incident on u; and (3) the coefficients $a_{i,j}$, $a_{i',j}$, b_i, and c_j are either universal constants or encoded in the local input $\mathsf{I}(u)$ of node $u \in V$ for every $i,i' \in [k]$ and $j \in [\ell]$ such that $v(i) = u$, $v(i') = u'$, and $e(j) = (u,u')$.

The *primal dual* method facilitates the design of an α-APLS, $\alpha = \beta\cdot\gamma$, for a (β,γ)-fitted and locally verifiable OptDGP $\Psi = \langle \Pi, f\rangle$ whose goal is to determine for a given IO graph $G_{\mathsf{I},\mathsf{O}}$ if the output assignment $\mathsf{O} : V \to \{0,1\}^*$ is an optimal (feasible) solution for the co-legal G and I or α-far from being an optimal solution. Let $\mathbf{x}$ be the primal variable vector encoded in the output assignment O. If O is an optimal solution for G and I, then the prover uses a sequential algorithm to generate a feasible dual variable vector $\mathbf{y}$ such that $\mathbf{x}$ and $\mathbf{y}$ meet the relaxed complementary slackness conditions with parameters β and γ (such a dual solution $\mathbf{y}$

exists as Ψ is (β, γ)-fitted). The label assignment $L : V \to \{0,1\}^*$ constructed by the prover assigns to each node $u \in V$, a label $L(u)$ that encodes the vector $\mathbf{y}(u) = \langle y_i \mid v(i) = u \rangle$ of dual variables mapped to u in the dual variable vector $\mathbf{y}$.

Consider some node $u \in V$ of the given IO graph $G_{\mathsf{I},\mathsf{O}}$. The verifier at node u extracts (i) the vector $\mathbf{x}(u) = \langle x_j \mid u \in e(j) \rangle$ of primal variables mapped to edges incident on u from the local output $\mathsf{O}(u)$; (ii) the vector $\mathbf{y}(u)$ of dual variables mapped to u from the label $L(v)$; (iii) the vector $\mathbf{y}^N(u) = \langle y_i \mid v(i) \in N(u) \rangle$ of dual variables mapped to u's neighbors from the label vector $L^N(u)$; and (iv) the vectors $\mathbf{a}(u) = \langle a_{i,j} \mid u \in e(j) \rangle$, $\mathbf{b}(u) = \langle b_i \mid v(i) = u \rangle$, and $\mathbf{c}(u) = \langle c_j \mid u \in e(j) \rangle$ of coefficients mapped to u and the edges incident on u from the local input $\mathsf{I}(u)$.

The verifier at node $u \in V$ then proceeds as follows: (1) using $\mathbf{x}(u)$, $\mathbf{a}(u)$, and $\mathbf{b}(u)$, the verifier verifies that the primal constraints that correspond to rows $i \in [k]$ such that $v(i) = u$ are satisfied; (2) using $\mathbf{y}(u)$, $\mathbf{y}^N(u)$, $\mathbf{a}(u)$, and $\mathbf{c}(u)$, the verifier verifies that the dual constraints that correspond to columns $j \in [\ell]$ such that $u \in e(j)$ are satisfied; (3) using $\mathbf{x}(u)$, $\mathbf{y}(u)$, $\mathbf{y}^N(u)$, $\mathbf{a}(u)$, and $\mathbf{c}(u)$, the verifier verifies that the primal relaxed complementary slackness conditions that correspond to primal variables x_j such that $u \in e(j)$ hold with parameter β; and (4) using $\mathbf{x}(u)$, $\mathbf{y}(u)$, $\mathbf{a}(u)$, and $\mathbf{b}(u)$, the verifier verifies that the dual relaxed complementary slackness conditions that correspond to dual variables y_i such that $v(i) = u$ hold with parameter γ. If all four conditions are satisfied, then the verifier at node u returns `True`; otherwise, it returns `False`. Put together, the verifier accepts the IO graph $G_{\mathsf{I},\mathsf{O}}$ if and only if $\mathbf{x}$ and $\mathbf{y}$ are feasible primal and dual solutions that satisfy the primal and dual relaxed complementary slackness conditions with parameters β and γ, respectively.

To establish the correctness of the α-APLS, notice first that the primal constraints are satisfied if and only if O is a feasible solution for G and I. Assuming that primal constraints are satisfied, if O is an optimal solution for G and I, then the fact that Ψ is (β, γ)-fitted implies that the verifier generates a feasible dual solution $\mathbf{y}$ such that the primal and dual relaxed complementary slackness conditions are satisfied with parameters β and γ. Conversely, If $\mathbf{y}$ is a feasible dual solution and the primal and dual relaxed complementary slackness conditions are satisfied with parameters β and γ, then $\mathbf{x}$ approximates the optimal primal (fractional) solution within an approximation bound of $\beta \cdot \gamma = \alpha$, hence O approximates $OPT_\Psi(G, \mathsf{I})$ within the same approximation bound.

The proof size of a $(\beta \cdot \gamma)$-APLS for a (β, γ)-fitted and locally verifiable OptDGP Ψ, designed by means of the primal dual method, is the maximum number of bits required to encode the vector $\mathbf{y}(u)$ of dual variables mapped to a node $u \in V$. Let r be the range of possible values assigned by the prover to a dual variable y_i. We aim for schemes that minimize r. Particularly, for OptDGPs where the number of primal constraints mapped to each node is bounded by a constant, this results in a $(\beta \cdot \gamma)$-APLS with a proof size of $O(\log r)$.

In [10] we present APLSs that are obtained using the primal dual method. We note that for all these APLSs, both the prover and verifier run in polynomial sequential time, thus yielding sequentially efficient APLSs.

3.2 The Verifiable Centralized Approximation Method

Consider some OptDGP $\Psi = \langle \Pi, f \rangle$. We say that Ψ is *identified* if the input assignment $\mathsf{I} : V \to \{0,1\}^*$ encodes a unique id represented using $O(\log n)$ bits at each node $v \in V$ (possibly among other input components) for every IO graph $G_{\mathsf{I},\mathsf{O}}$.

We say that Ψ is *decomposable* if there exists a function $\lambda : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$, often referred to as a *decomposition function*, such that $f(G_{\mathsf{I},\mathsf{O}}) = \sum_{v \in V} \lambda(\mathsf{I}(v), \mathsf{O}(v))$ for every IO graph $G_{\mathsf{I},\mathsf{O}} \in \Pi$ (cf. the notion of semi-group functions in [26]). Given an input and

output assignments $\mathsf{I}, \mathsf{O} : V \to \{0,1\}^*$, let $\lambda(\mathsf{I}, \mathsf{O}) = \sum_{v \in V} \lambda(\mathsf{I}(v), \mathsf{O}(v))$ denote the sum of the decomposition function values $\lambda(\mathsf{I}(v), \mathsf{O}(v))$ over all nodes $v \in V$. Notice that the decomposition function is well defined for all bit string pairs; in particular, the definition of $\lambda(\mathsf{I}, \mathsf{O})$ does not require that the output assignment O is a feasible solution for the graph G and the input assignment I.

Let $\Psi = \langle \Pi, f \rangle$ be a decomposable MinDGP (resp., MaxDGP) with a decomposition function $\lambda : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$. Given a legal input graph $G_{\mathsf{I}} \in \Pi$ and a parameter $\alpha \geq 1$, we say that a (not necessarily feasible) output assignment $\mathsf{A} : V \to \{0,1\}^*$ is a *decomposable α-approximation* for G and I if $OPT_\Psi(G, \mathsf{I}) \leq \lambda(\mathsf{I}, \mathsf{A}) \leq \alpha \cdot OPT_\Psi(G, \mathsf{I})$ (resp., $OPT_\Psi(G, \mathsf{I})/\alpha \leq \lambda(\mathsf{I}, \mathsf{A}) \leq OPT_\Psi(G, \mathsf{I})$).

Fix some identified decomposable MinDGP (resp., MaxDGP) $\Psi = \langle \Pi, f \rangle$ with a decomposition function $\lambda : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$. The *verifiable centralized approximation (VCA)* method facilitates the design of an α-ADPLS for Ψ whose goal is to determine for a given legal input graph $G_{\mathsf{I}} \in \Pi$ and some parameter $k \in \mathbb{Z}$ if every output assignment yields an objective value of at least (resp., at most) k or if there exists an output assignment that yields an objective value smaller than k/α (resp., larger than $\alpha \cdot k$). The ADPLSs designed by means of the VCA method are composed of two verification tasks, namely, the *approximation* task and the *comparison* task, so that the verifier accepts G_{I} if and only if both verification tasks accept. The label $L(v) = \langle L_{\texttt{approx}}(v), L_{\texttt{comp}}(v) \rangle$ assigned by the prover to each node $v \in V$ is composed of the fields $L_{\texttt{approx}}(v)$ and $L_{\texttt{comp}}(v)$ serving the approximation task and the comparison task, respectively.

In the approximation task, the prover runs a centralized algorithm `ALG` that is guaranteed to produce a decomposable α-approximation $\mathsf{A} : V \to \{0,1\}^*$ for graph G and input assignment I. The $L_{\texttt{approx}}(v)$ field of the label $L(v)$ assigned by the prover to each node $v \in V$ consists of both $\mathsf{A}(v)$ and a proof that the output assignment A is indeed the outcome of the centralized algorithm `ALG`. The correctness requirement for this task is defined so that the verifier accepts G_{I} if and only if the field $L_{\texttt{approx}}(v)$ encodes an output assignment A that can be obtained using `ALG`.

The purpose of the comparison task is to verify that $\lambda(\mathsf{I}, \mathsf{A}) \geq k$ (resp., $\lambda(\mathsf{I}, \mathsf{A}) \leq k$), where λ is the decomposition function associated with the (decomposable) MinDGP (resp., MaxDGP) Ψ and A is the output assignment encoded in the $L_{\texttt{approx}}(v)$ fields of the labels $L(v)$ assigned to nodes $v \in V$. This is done by means of the (λ, k)-comparison scheme (resp., the $(-\lambda, -k)$-comparison scheme) presented in Section 2.

The correctness of the α-ADPLS for the MinDGP (resp., MaxDGP) Ψ and the integer k is established as follows. If $OPT_\Psi(G, \mathsf{I}) \geq k$ (resp., $OPT_\Psi(G, \mathsf{I}) \leq k$), then the $L_{\texttt{approx}}(v)$ field of the label $L(v)$ assigned by the prover to each node $v \in V$ encodes an output assignment $\mathsf{A} : V \to \{0,1\}^*$ generated by the algorithm `ALG`. This means that A is a decomposable α-approximation, thus $\lambda(\mathsf{I}, \mathsf{A}) \geq OPT_\Psi(G, \mathsf{I}) \geq k$ (resp., $\lambda(\mathsf{I}, \mathsf{A}) \leq OPT_\Psi(G, \mathsf{I}) \leq k$) and the verifier accepts G_{I}. On the other hand, if $OPT_\Psi(G, \mathsf{I}) < k/\alpha$ (resp., $OPT_\Psi(G, \mathsf{I}) > \alpha \cdot k$), then for any decomposable α-approximation A, it holds that $\lambda(\mathsf{I}, \mathsf{A}) \leq \alpha \cdot OPT_\Psi(G, \mathsf{I}) < k$ (resp., $\lambda(\mathsf{I}, \mathsf{A}) \geq OPT_\Psi(G, \mathsf{I})/\alpha > k$), hence the verifier rejects G_{I} for any label assignment $L : V \to \{0,1\}^*$.

The proof size of the α-ADPLS designed via the VCA method is the maximum size of a label $L(v) = \langle L_{\texttt{approx}}(v), L_{\texttt{comp}}(v) \rangle$ assigned by the prover for a given input graph G_{I} such that $OPT_\Psi(G, \mathsf{I}) \geq k$ (resp., $OPT_\Psi(G, \mathsf{I}) \leq k$). As discussed in Section 2, it is guaranteed that the $L_{\texttt{comp}}(\cdot)$ fields are represented using $O(\log n + H)$ bits, where H is an upper bound on the number of bits required to represent a $\lambda(\mathsf{I}(v), \mathsf{O}(v))$ value for any $v \in V$, and A is the decomposable α-approximation generated by the prover in the approximation task. In [10], we develop ADPLSs whose $L_{\texttt{approx}}(\cdot)$ fields are also represented using $|L_{\texttt{approx}}(v)| = O(\log n + H)$

bits. Moreover, the OptDGPs we consider admit some fixed parameter $W \in \mathbb{Z}$ (typically an upper bound on the weights in the graph) such that $H = O(\log n + \log W)$ which results in a proof size of $O(\log n + \log W)$.

A desirable feature of the ADPLSs we develop in [10] is that the centralized algorithms $\texttt{ALG}$ employed in the approximation task are efficient, hence the prover runs in polynomial time. Since the (sequential) runtime of the verifier is also polynomial, it follows that all our ADPLSs are sequentially efficient.

4 Reductions Between APLSs and ADPLSs

4.1 From an α-ADPLS to an α-APLS

Consider an identified decomposable MinDGP (resp., MaxDGP) $\Psi = \langle \Pi, f \rangle$ with a decomposition function $\lambda : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$. Let p and r be the proof sizes of a feasibility scheme for Π and an α-ADPLS for Ψ, respectively. We establish the following lemma.

▶ **Lemma 4.1.** *There exists an α-APLS for Ψ with a proof size of $O(p + r + \log n + H)$, where H is the maximal number of bits required to represent $\lambda(\mathsf{I}(v), \mathsf{O}(v))$ for any $v \in V$.*

Proof. Observe that if O is known to be a feasible solution for G and I, then the correctness requirements of an α-APLS for the MinDGP (resp., MaxDGP) Ψ are equivalent to those of an α-ADPLS for Ψ and $k = f(G_{\mathsf{I},\mathsf{O}})$. That is, for a given IO graph $G_{\mathsf{I},\mathsf{O}} \in \Pi$, if $OPT_\Psi(G, \mathsf{I}) \geq k$ (resp., $OPT_\Psi(G, \mathsf{I}) \leq k$), then O is an optimal solution for G and I which requires the verifier of an α-APLS to accept $G_{\mathsf{I},\mathsf{O}}$; if $OPT_\Psi(G, \mathsf{I}) < k/\alpha$ (resp., $OPT_\Psi(G, \mathsf{I}) > \alpha \cdot k$), then O is at least α-far from being optimal for G and I which requires the verifier of an α-APLS to reject $G_{\mathsf{I},\mathsf{O}}$.

The design of an α-APLS for Ψ is thus enabled by taking the label assigned by the prover to each node $v \in V$ to be $L(v) = \langle L_{\texttt{feas}}(v), L_{\texttt{obj}}(v), L_{\texttt{comp}}(v), L_{\texttt{ADPLS}}(v) \rangle$, where $L_{\texttt{feas}}(v)$ is the p-bit label assigned to v by the prover of the feasibility scheme for Π; $L_{\texttt{obj}}(v) = f(G_{\mathsf{I},\mathsf{O}})$ (note that all nodes are assigned with the same $L_{\texttt{obj}}(\cdot)$ field); $L_{\texttt{comp}}(v)$ is the label constructed in the $(\lambda, L_{\texttt{obj}}(v))$-comparison scheme (resp., the $(-\lambda, -L_{\texttt{obj}}(v))$-comparison scheme) presented in Section 2; and $L_{\texttt{ADPLS}}(v)$ is the r-bit label of an α-ADPLS for Ψ and $L_{\texttt{obj}}(v)$. This label assignment allows the verifier to verify that (1) O is a feasible solution for G and I; (2) $f(G_{\mathsf{I},\mathsf{O}}) \geq L_{\texttt{obj}}(v)$ (resp., $f(G_{\mathsf{I},\mathsf{O}}) \leq L_{\texttt{obj}}(v)$) for each $v \in V$; and (3) the verifier of an α-ADPLS for Ψ and $k = f(G_{\mathsf{I},\mathsf{O}})$ accepts the input graph G_{I}. ◀

4.2 From an α-APLS to an α-ADPLS

Consider an identified, locally verifiable, and (β, γ)-fitted OptDGP Ψ with the mappings $v : [s] \to V$ and $e : [t] \to E$ that are associated with its LP relaxation P whose matrix form is given by the variable vector $\mathbf{x} = \{x_j\} \in \mathbb{R}^t$ and coefficient matrix and vectors $\mathbf{A} = \{a_{i,j}\} \in \mathbb{R}^{s \times t}$, $\mathbf{b} = \{b_i\} \in \mathbb{R}^s$, and $\mathbf{c} = \{c_j\} \in \mathbb{R}^t$. Define $D_u = \{i \mid v(i) = u\}$ for each $u \in V$ and let $d = \max_{u \in V}\{|D_u|\}$. Let $b_{\max}$ be the maximal number of bits required to represent b_i for any $i \in [s]$. Let $\alpha = \beta \cdot \gamma$ and let r be the proof size of an α-APLS for Ψ produced by the primal dual method. We obtain the following lemma.

▶ **Lemma 4.2.** *There exists an α-ADPLS for Ψ with a proof size of $O(\log n + \log d + b_{\max} + r)$.*

Proof. We construct an α-ADPLS for the MinDGP (resp., MaxDGP) Ψ by means of the VCA method. Recall that an α-APLS for Ψ established by means of the primal dual method is defined so that the labels encode a feasible dual solution $\mathbf{y} \in \mathbb{R}^s$ that satisfies $OPT_\Psi(G, \mathsf{I}) \leq \alpha \cdot \mathbf{b}^{\mathrm{T}}\mathbf{y}$ (resp., $OPT_\Psi(G, \mathsf{I}) \geq \frac{1}{\alpha} \cdot \mathbf{b}^{\mathrm{T}}\mathbf{y}$). Define $\mathsf{A}(u) = \mathbf{y}(u) = \langle y_i \mid v(i) = u \rangle$

and $\lambda(\mathsf{I}(u), \mathsf{A}(u)) = \alpha \cdot \sum_{i:v(i)=u} b_i y_i$ (resp., $\lambda(\mathsf{I}(u), \mathsf{A}(u)) = \frac{1}{\alpha} \cdot \sum_{i:v(i)=u} b_i y_i$) for each $u \in V$. The prover sets the sub-label $L_{\texttt{approx}}(u) = \mathsf{A}(u) = \mathbf{y}(u)$ associated with the approximation task for each node $u \in V$, which allows the verifier to verify that $\mathbf{y}$ is a feasible dual solution.

For the correctness of this scheme, it suffices to show that A is a decomposable α-approximation for G and I (with respect to the decomposition function λ). Note that $\mathbf{y}$ is defined so that it satisfies $OPT_\Psi(G, \mathsf{I}) \leq \alpha \cdot \mathbf{b}^{\mathrm{T}}\mathbf{y}$ (resp., $OPT_\Psi(G, \mathsf{I}) \geq \frac{1}{\alpha} \cdot \mathbf{b}^{\mathrm{T}}\mathbf{y}$); and weak duality implies that $\alpha \cdot \mathbf{b}^{\mathrm{T}}\mathbf{y} \leq \alpha \cdot OPT_\Psi(G, \mathsf{I})$ (resp., $\frac{1}{\alpha} \cdot \mathbf{b}^{\mathrm{T}}\mathbf{y} \geq \frac{1}{\alpha} \cdot OPT_\Psi(G, \mathsf{I})$). It follows that A is a decomposable α-approximation for G and I since $\lambda(\mathsf{I}, \mathsf{A}) = \sum_{u \in V} \lambda(\mathsf{I}(u), \mathsf{A}(u)) = \alpha \cdot \sum_{u \in V} \sum_{i:v(i)=u} b_i y_i = \alpha \cdot \mathbf{b}^{\mathrm{T}}\mathbf{y}$ (resp., $\lambda(\mathsf{I}, \mathsf{A}) = \sum_{u \in V} \lambda(\mathsf{I}(u), \mathsf{A}(u)) = \frac{1}{\alpha} \cdot \sum_{u \in V} \sum_{i:v(i)=u} b_i y_i = \frac{1}{\alpha} \cdot \mathbf{b}^{\mathrm{T}}\mathbf{y}$). ◀

References

1. B. Awerbuch, B. Patt-Shamir, and G. Varghese. Self-stabilization by local checking and correction. In *Proceedings 32nd Annual Symposium of Foundations of Computer Science*, pages 268–277, 1991.
2. Nir Bacrach, Keren Censor-Hillel, Michal Dory, Yuval Efron, Dean Leitersdorf, and Ami Paz. Hardness of distributed optimization. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019.
3. Alkida Balliu, Gianlorenzo D'Angelo, Pierre Fraigniaud, and Dennis Olivetti. What can be verified locally? *J. Comput. Syst. Sci.*, 97:106–120, 2018.
4. Lélia Blin, Pierre Fraigniaud, and Boaz Patt-Shamir. On proof-labeling schemes versus silent self-stabilizing algorithms. In *Stabilization, Safety, and Security of Distributed Systems*, pages 18–32, 2014.
5. Keren Censor-Hillel, Ami Paz, and Mor Perry. Approximate proof-labeling schemes. *Theor. Comput. Sci.*, 811:112–124, 2020.
6. Miroslav Chlebík and Janka Chlebíková. The steiner tree problem on graphs: Inapproximability results. *Theor. Comput. Sci.*, 406(3):207–214, 2008.
7. Pierluigi Crescenzi, Pierre Fraigniaud, and Ami Paz. Trade-offs in distributed interactive proofs. In *33rd International Symposium on Distributed Computing*, pages 13:1–13:17, 2019.
8. Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005.
9. Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Symposium on Theory of Computing, STOC*, pages 624–633, 2014.
10. Yuval Emek and Yuval Gil. Twenty-two new approximate proof labeling schemes (full version), 2020. `arXiv:2007.14307`.
11. Laurent Feuilloley. Introduction to local certification, 2019.
12. Laurent Feuilloley and Pierre Fraigniaud. Survey of distributed decision. *Bull. EATCS*, 119, 2016.
13. Laurent Feuilloley and Pierre Fraigniaud. Error-sensitive proof-labeling schemes. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, volume 91 of *LIPIcs*, pages 16:1–16:15, 2017.
14. Laurent Feuilloley, Pierre Fraigniaud, Juho Hirvonen, Ami Paz, and Mor Perry. Redundancy in distributed proofs. In *32nd International Symposium on Distributed Computing, DISC*, volume 121 of *LIPIcs*, pages 24:1–24:18, 2018.
15. Pierre Fraigniaud, Amos Korman, and David Peleg. Local distributed decision. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS*, pages 708–717, 2011.
16. Pierre Fraigniaud, Pedro Montealegre, Rotem Oshman, Ivan Rapaport, and Ioan Todinca. On distributed merlin-arthur decision protocols. In *Structural Information and Communication Complexity*, pages 230–245, 2019.
17. Pierre Fraigniaud, Boaz Patt-Shamir, and Mor Perry. Randomized proof-labeling schemes. *Distributed Comput.*, 32(3):217–234, 2019.

18 Mika Göös and Jukka Suomela. Locally checkable proofs in distributed computing. *THEORY OF COMPUTING*, 12:1–33, 2016.

19 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, July 2001.

20 Marek Karpinski, Michael Lampis, and Richard Schmied. New inapproximability bounds for TSP. *J. Comput. Syst. Sci.*, 81(8):1665–1677, 2015.

21 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, page 767–775, 2002.

22 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing*, 37(1):319–357, 2007.

23 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008.

24 Gillat Kol, Rotem Oshman, and Raghuvansh R. Saxena. Interactive distributed proofs. In *PODC 2018 - Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 255–264, July 2018.

25 Amos Korman and Shay Kutten. Distributed verification of minimum spanning trees. *Distributed Comput.*, 20(4):253–266, 2007.

26 Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Comput.*, 22(4):215–233, 2010.

27 E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 2006.

28 Moni Naor, Merav Parter, and Eylon Yogev. The power of distributed verifiers in interactive proofs. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1096–115, 2020.

29 Rafail Ostrovsky, Mor Perry, and Will Rosenbaum. Space-time tradeoffs for distributed verification. In Shantanu Das and Sebastien Tixeuil, editors, *Structural Information and Communication Complexity*, pages 53–70, Cham, 2017. Springer International Publishing.

30 Boaz Patt-Shamir and Mor Perry. Proof-labeling schemes: Broadcast, unicast and in between. In *Stabilization, Safety, and Security of Distributed Systems - 19th International Symposium, SSS*, volume 10616, pages 1–17. Springer, 2017.