Uniform-Circuit and Logarithmic-Space Approximations of Refined Combinatorial Optimization Problems

Tomoyuki Yamakami

Department of Information Science, University of Fukui
3-9-1 Bunkyo, Fukui, 910-8507 Japan

Abstract. We lay out a refined framework to discuss various approximation algorithms for combinatorial optimization problems residing inside the optimization class PO. We are focused on optimization problems characterized by computation models of uniform NC^1-circuits, uniform-AC^0, and logarithmic-space Turing machines. We present concrete optimization problems and prove that they are indeed complete under reasonably weak reductions. We also show collapses and separations among refined optimization classes.

Keywords: optimization problem, approximation-preserving reduction, approximation algorithm, NC^1 circuit, AC^0 circuit, logarithmic space.

1 Introduction

A *combinatorial optimization problem* asks to find an "optimal" solution among all feasible solutions associated with each admissible instance, where the optimality usually takes a form of either maximization or minimization according to a certain fixed ordering over all solutions. A significant progress was made in a field of fundamental research during 1990s and its trend has continued promoting our understandings of the approximability of optimization problems. In particular, *NP optimization problems* (or *NPO problems*, in short) have been a centerfold of our interests in a direct connection to NP decision problems. Let NPO express the collection of such optimization problems. NPO problems that can be exactly solved in polynomial time form a "tractable" optimization class PO, whereas APX (which is denoted in this paper by APXP for technicality) consists of NPO problems whose optimum solutions are relatively approximated within constant factors in polynomial time. A large number of NPO problems that have been studied are classified into those complexity classes.

Those classifications of optimization problems are all described from a viewpoint of polynomial-time computability and any systematic discussion on optimization problems inside PO has been vastly neglected except for [7], in which *logarithmic-space optimization problems* (or *NLO problems*) were discussed. Note that Àlvarez and Jenner [1] also studied from a slightly different viewpoint a class OptL of functions computing optimal solutions using only logarithmic space.

P. Widmayer, Y. Xu, and B. Zhu (Eds.): COCOA 2013, LNCS 8287, pp. 318–329, 2013.

A number of intriguing optimization problems have been already known to reside within PO. As a typical example, the problem, MIN ST-CUT, of finding a minimal *s-t* cut of a given directed graph is well-known to belong to PO. Another example is the *minimum path weight problem* (MIN PATH-WEIGHT), which is to find a path $\mathcal{S} = (v_1, v_2, \ldots, v_k)$ with $k \geq 2$ from given source v_1 to sink v_k on a given directed graph such that the path weight $\sum_{i=1}^{k} w(v_i)$ is minimum, where $w(v)$ is a given weight (expressed in binary) of vertex v. This problem is also in PO. Although MIN ST-CUT is a "complete" problem for PO, MIN PATH-WEIGHT, which is actually an NLO problem, seems to have low complexity within PO.

Here, we wish to raise a question of whether it is possible to obtain a finer classification inside PO. To achieve our goal, we seek to develop a *new, finer framework*—a low-complexity world of optimization problems—and reexamine the computational complexity of optimization problems within this new framework. In particular, we look into a world of optimization problems that can be approximated by *logarithmic-space* (or *log-space*) Turing machines and by uniform families of NC^1*-circuits*. For this purpose, we need to reshape the existing framework of expressing optimization complexity classes by clarifying the scope and complexity of verification processes for solutions and objective functions. For instance, to expand the approximation class APX into lower complexity classes, we intend to use a new notation $\mathrm{APXP_{NPO}}$ to emphasize the polynomial-time approximability of NP optimization problems. Similarly, we describe a collection of NP optimization problems that are P-solvable as $\mathrm{PO_{NPO}}$.

As logarithmic-space computation has often exhibited intriguing features in the past decades, significant differences also exist between NPO and NLO. For example, unlike NPO problems, weak computation models do not seem to support a typical reduction between minimization problems and maximization problems within NLO unless they are polynomially bounded (see Section 3.2).

The optimization class $\mathrm{LO_{NLO}}$ was introduced in [7] as a collection of NL optimization problems that are L-solvable (i.e., solvable by multi-tape deterministic Turing machines using logarithmic space). If we replace directed graphs of MIN PATH-WEIGHT by undirected forests, then the resulted problem, called MIN FOREST-PATH-WEIGHT, belongs to $\mathrm{LO_{NLO}}$. In a similar way, using log-space uniform families of NC^1-circuits and AC^0-circuits in place of logarithmic-space Turing machines, we can define $\mathrm{NC}^1\mathrm{O_{NLO}}$ and $\mathrm{AC}^0\mathrm{O_{NLO}}$, respectively.

We will present a number of concrete optimization problems that are "complete" for the aforementioned refined classes of optimization problems under weak reductions. We need such weak reductions among low-complexity optimization problems because strong reductions tend to obscure essential characteristics of "complete" problems. We will also prove relationships among those classes.

2 Optimization and Approximation Preliminaries

We will refine an existing framework for studying combinatorial optimization problems of, in particular, low computational complexity. Throughout this paper, the notation $\mathbb{N}$ denotes the set of all *natural numbers* (i.e., nonnegative

integers) and $\mathbb{Q}$ indicates the set of all *rational numbers*. Two special notations $\mathbb{Q}^{>1}$ and $\mathbb{Q}^{\geq 1}$ respectively express the sets $\{q \in \mathbb{Q} \mid q > 1\}$ and $\{q \in \mathbb{Q} \mid q \geq 1\}$. Given two numbers $m, n \in \mathbb{N}$ with $m \leq n$, an *integer interval* $[m, n]_{\mathbb{Z}}$ is a set $\{m, m+1, m+2, \ldots, n\}$. A *string* (or a *word*) over alphabet Σ is a finite series of symbols taken from Σ. The *empty string* is denoted λ. Given a binary string w, $rep(w)$ denotes the positive integer represented by w in binary.

2.1 Models of Computation

As a model of computation, we will use the following basic form of *Turing machine*, which is equipped with a *random-access* input tape, an input-index tape, multiple work tapes, and possibly an output tape. A tape is called *read-once* if it is a read-only tape and its tape head either stays at the same cell without reading any information (whose move is called an λ-move or ε-move) or moves to the right cell to scan another symbol. Similarly, a *write-only* tape indicates that, whenever its tape head writes a nonempty symbol in a tape cell, the head should move immediately to its right tape cell. In this paper, "output tapes" are always assumed to be write-only tapes.

An *auxiliary Turing machine* is the above-mentioned deterministic Turing machine equipped with an extra read-once *auxiliary tape* on which a sequence of symbols is provided as an extra input. This machine can therefore read off two symbols at once from an input tape and an auxiliary tape to make a deterministic move. Let auxL denotes the collection of all sets A for which there exist a polynomial p and a log-space auxiliary Turing machine M such that, for every x and y, (i) $(x, y) \in A$ implies $|y| \leq p(|x|)$ and (ii) M accepts (x, y) iff $(x, y) \in A$, where y is given on an auxiliary tape. Its functional version (with polynomially-bounded output symbols) is denoted by auxFL.

We assume that the reader is familiar with four complexity classes, P, NP, L, and NL, and two function classes, FP and FL. For circuit-based complexity classes AC^0 and NC^1 (and their functional versions FAC^0 and FNC^1), we use a standard notion of *Boolean circuits*, which are composed only of three basic gates AND, OR, and NOT. A family of NC^1-circuits requires *log-space uniformity*, whereas a family of AC^0-circuits requires *DLOGTIME-uniformity*.

It is important to note that, on an output tape of a machine, a natural number is represented in binary, where the least significant bit is always placed at the right end of the output bits.

2.2 Refined Optimization Classes

Combinatorial optimization problems that we will extensively discuss in this paper can be formulated in the following manner. Since our purpose is to examine lower-complexity problems, it is better to reformulate an existing framework of *NP optimization problems* or *NPO problems* (see, e.g., [2]) in terms of auxiliary Turing machines.

An NPO problem $P = (I, SOL, m, goal)$:

- I is a finite set of *admissible instances*. There must be a deterministic Turing machine (DTM) that recognizes I in polynomial time.
- SOL is a function mapping I to a collection of certain finite sets, where $SOL(x)$ is a set of *feasible solutions* of input instance x. There must be a polynomial q such that (i) for every $x \in I$ and every $y \in SOL(x)$, it holds that $|y| \leq q(|x|)$ and (ii) the set $I \circ SOL = \{(x, y) \mid x \in I, y \in SOL(x)\}$ is recognized in time polynomial in $|x|$ by a certain auxiliary Turing machine stating with x on an input tape and y on an auxiliary tape.
- *goal* is either MAX or MIN. When *goal* = MAX, P is called a *maximization problem*; when *goal* = MIN, it is a *minimization problem*.
- m is a *measure function* (or an *objective function*) from $I \circ SOL$ to $\mathbb{N}$ whose value $m(x, y)$ is computed in time polynomial in $|x|$ by a certain auxiliary Turing machine starting with x on an input tape and y on an auxiliary tape. For any instance $x \in I$, $m^*(x)$ denotes the "optimal" value $goal\{m(x, y) \mid y \in SOL(x)\}$. Moreover, $SOL^*(x)$ expresses the "optimal" set $\{y \in SOL(x) \mid m(x, y) = m^*(x)\}$ of x.

Since a polynomial-time Turing machine can copy y into its work tape and manipulate it freely, the above use of auxiliary Turing machines does not alter the existing notion of NPO problems. Let the notation NPO express the class of all NPO problems. We say that an NPO problem P is P-*solvable* if there exists a polynomial-time deterministic algorithm M such that, for every instance $x \in I$, M returns an optimal solution y in $SOL(x)$ (possibly together with its optimal value $m^*(x)$). To analyze log-space optimization problems, Tantau [7] considered *NL optimization problems* (or *NLO problems*, in short), which are obtained simply by replacing the term "polynomial time" in the above definition of NPO problems with "logarithmic space." For NLO problems, the use of auxiliary Turing machine is essential and it may not be replaced by any Turing machine having no read-once tapes. To express the class of all NLO problems, we use the succinct notation of NLO. Moreover, MinNL (MaxNL, resp.) denotes the class of all minimization (maximization, resp.) problems in NLO. Given a class $\mathcal{C}$ of optimization problems, the notation $\mathrm{PO}_{\mathcal{C}}$ expresses the class of all optimization problems in $\mathcal{C}$ that are P-solvable. Similarly, we can define the notions of $\mathrm{LO}_{\mathcal{C}}$, $\mathrm{NC}^1\mathrm{O}_{\mathcal{C}}$, and $\mathrm{AC}^0\mathrm{O}_{\mathcal{C}}$ by replacing the term "P-solvable" with "L-solvable," "NC^1-solvable," and "AC^0-solvable," respectively. Conventionally, $\mathrm{PO}_{\mathrm{NPO}}$ is written as PO, and $\mathrm{LO}_{\mathrm{NLO}}$ is noted briefly as LO in [7].

A measure function m is called *polynomially bounded* if there exists a polynomial p such that $m(x, y) \leq p(|x|, |y|)$ holds for all pairs $(x, y) \in I \circ SOL$. Moreover, an optimization problem is said to be *polynomially bounded* if its measure function is polynomially bounded. We use a succinct notation PBO to denote the collection of all optimization problems that are polynomially bounded.

Next, we will define approximation classes using a notion of γ-approximation. Given an optimization problem $P = (I, SOL, m, goal)$, the *performance ratio* of solution y with respect to instance x is defined as $R(x, y) = \max\{|m(x, y)/m^*(x)|, |m^*(x)/m(x, y)|\}$, provided that neither $m(x, y)$ nor $m^*(x)$ is zero. Notice that $R(x, y) = 1$ iff $y \in SOL^*(x)$. Let $\gamma > 1$ be a

constant indicating an upper bound of performance ratio. We say that P is *polynomial-time γ-approximable* if there exists a polynomial-time deterministic Turing machine M such that, for any instance x, $R(x, M(x)) \leq \gamma$. Such a machine is also called a *γ-approximate algorithm*. The γ-approximability implies that the set $\{x \in I \mid SOL(x) \neq \emptyset\}$ is in P. We also define three extra notions of "log-space γ-approximation" [7], "NC^1 γ-approximation," and "AC^0 γ-approximation" by replacing "polynomial-time Turing machine" in the above definition with "logarithmic-space (auxiliary) Turing machine," "uniform family of NC^1-circuits," and "uniform family of AC^0-circuits," respectively.

The notation $\mathrm{APXP}_{\mathcal{C}}$ denotes a class consisting of problems P in class $\mathcal{C}$ of optimization problems such that, for a certain fixed constant $\gamma > 1$, P is polynomial-time γ-approximable. Similarly, we introduce the notations of $\mathrm{APXL}_{\mathcal{C}}$, $\mathrm{APXNC}^1_{\mathcal{C}}$, and $\mathrm{APXAC}^0_{\mathcal{C}}$ using "log-space γ-approximation," "NC^1 γ-approximation," and "AC^0 γ-approximation," respectively. Notice that $\mathrm{APXP}_{\mathrm{NPO}}$ is conventionally expressed as APX.

2.3 Approximation-Preserving Reductions

We will use three types of reductions between two optimization problems. Given two optimization problems $P = (I_1, SOL_1, m_1, goal)$ and $Q = (I_2, SOL_2, m_2, goal)$, P is *polynomial-time AP-reducible* (or *APP-reducible*, in short) to Q, denoted $P \leq^{\mathrm{P}}_{\mathrm{AP}} Q$, if there are two functions f, g and a constant $c \geq 1$ such that the following *APP-condition* is satisfied:

- for any instance $x \in I_1$ and any $r \in \mathbb{Q}^{>1}$, it holds that $f(x, r) \in I_2$,
- for any $x \in I_1$ and any $r \in \mathbb{Q}^{>1}$, if $SOL_1(x) \neq \emptyset$ then $SOL_2(f(x, r)) \neq \emptyset$,
- for any $x \in I_1$, any $r \in \mathbb{Q}^{>1}$, and any $y \in SOL_2(f(x, r))$, it holds that $g(x, y, r) \in SOL_1(x)$,
- $f(x, r)$ and $g(x, y, r)$ are computed by two deterministic auxiliary Turing machines that run in time polynomial in $(|x|, |y|)$ for any fixed $r \in \mathbb{Q}^{>1}$, and
- for any $x \in I_1$, any $r \in \mathbb{Q}^{>1}$, and any $y \in SOL_2(f(x, r))$, $R_2(f(x, r), y) \leq r$ implies $R_1(x, g(x, y, r)) \leq 1 + c(r - 1)$.

When the above APP-condition holds, we also say that P *APP-reduces* to Q. The triplet (f, g, c) is called a *polynomial-time AP-reduction* (or an *APP-reduction*) from P to Q.

Notice that the above definition excludes the case of $r = 1$. As a result, $\mathrm{PO}_{\mathrm{NPO}}$ is not closed under polynomial-time AP-reductions. Since our main target is problems inside $\mathrm{PO}_{\mathrm{NPO}}$, we further need to introduce another type of reduction (f, g), in which g "exactly" transforms in polynomial time an optimal solution for Q to another optimal solution for P. We write $P \leq^{\mathrm{P}}_{\mathrm{EX}} Q$ when the following *EX-condition* holds:

- for any instance $x \in I_1$, it holds that $f(x) \in I_2$,
- for any $x \in I_1$, if $SOL_1(x) \neq \emptyset$ then $SOL_2(f(x)) \neq \emptyset$,
- for any $x \in I_1$ and any $y \in SOL_2(f(x))$, it holds that $g(x, y) \in SOL_1(x)$,
- $f(x)$ and $g(x, y)$ are computed by two deterministic auxiliary Turing machines that run in time polynomial in $(|x|, |y|)$, and

- for any $x \in I_1$ and $y \in SOL_2(f(x))$, $R_2(f(x), y) = 1$ implies $R_1(x, g(x, y)) = 1$, where R_1 and R_2 respectively express performance ratios for P_1 and P_2.

The above pair (f, g) is called a *polynomial-time EX-reduction* from P to Q.

By combining $\leq^{\mathrm{P}}_{\mathrm{AP}}$ and $\leq^{\mathrm{P}}_{\mathrm{EX}}$, we define the third notion of *polynomial-time strong AP-reduction* (or strong APP-reduction), denoted $\leq^{\mathrm{P}}_{\mathrm{sAP}}$, obtained from $\leq^{\mathrm{P}}_{\mathrm{AP}}$ by allowing r to be chosen from $\mathbb{Q}^{\geq 1}$ (instead of $\mathbb{Q}^{>1}$).

By replacing the requirement of "polynomial time" in the above (strong) APP-condition with "logarithmic-space," "uniform family of NC^1-circuits," and "uniform family of AC^0-circuits," we obtain *(strong) APL-reduction* ($\leq^{\mathrm{L}}_{\mathrm{AP}}$, $\leq^{\mathrm{L}}_{\mathrm{sAP}}$), *(strong) APNC^1-reduction* ($\leq^{\mathrm{NC}^1}_{\mathrm{AP}}$, $\leq^{\mathrm{NC}^1}_{\mathrm{sAP}}$), and *(strong) APAC^0-reduction* ($\leq^{\mathrm{AC}^0}_{\mathrm{AP}}$, $\leq^{\mathrm{AC}^0}_{\mathrm{sAP}}$), respectively. The following lemma is immediate.

Lemma 1. *For any reduction type $c \in \{\mathrm{P}, \mathrm{L}, \mathrm{NC}^1, \mathrm{AC}^0\}$, $P_1 \leq^{c}_{\mathrm{sAP}} P_2$ implies both $P_1 \leq^{c}_{\mathrm{AP}} P_2$ and $P_1 \leq^{c}_{\mathrm{EX}} P_2$.*

Given a type of reduction, say, $\leq$ discussed above as well as a class $\mathcal{C}$ of optimization problems, an optimization problem P is called *$\leq$-hard* for $\mathcal{C}$ if $Q \leq P$ holds for every problem Q in $\mathcal{C}$. Moreover, P is said to be *$\leq$-complete* for $\mathcal{C}$ if P is in $\mathcal{C}$ and it is $\leq$-hard for $\mathcal{C}$.

3 Complete Problems

In Section 2, we have introduced basic classes of low-complexity optimization problems. Note that, for any given class $\mathcal{C}$ of optimization problems, $\mathrm{NC}^1\mathrm{O}_{\mathcal{C}} \subseteq \mathrm{LO}_{\mathcal{C}} \subseteq \mathrm{PO}_{\mathcal{C}}$ and $\mathrm{APXNC}^1_{\mathcal{C}} \subseteq \mathrm{APXL}_{\mathcal{C}} \subseteq \mathrm{APXP}_{\mathcal{C}}$. Moreover, it holds that $\mathrm{NC}^1\mathrm{O}_{\mathcal{C}} \subseteq \mathrm{APXNC}^1_{\mathcal{C}}$, $\mathrm{LO}_{\mathcal{C}} \subseteq \mathrm{APXL}_{\mathcal{C}}$, and $\mathrm{PO}_{\mathcal{C}} \subseteq \mathrm{APXP}_{\mathcal{C}}$.

3.1 General Complete Problems

Hereafter, we will discuss complete problems for refined optimization classes. We first note that the type of reduction is often crucial. The $\leq^{\mathrm{L}}_{\mathrm{AP}}$- and $\leq^{\mathrm{L}}_{\mathrm{EX}}$-reductions are quite powerful so that all problems in $\mathrm{APXL}_{\mathrm{NLO}}$ and $\mathrm{LO}_{\mathrm{NLO}}$ become reducible to problems even in $\mathrm{APXAC}^0_{\mathrm{NLO}}$ and $\mathrm{AC}^0\mathrm{O}_{\mathrm{NLO}}$, respectively.

Proposition 1. *1.* $\mathrm{APXL}_{\mathrm{NLO}} = \{P \in \mathrm{NLO} \mid \exists Q \in \mathrm{APXAC}^0_{\mathrm{NLO}}\, [P \leq^{\mathrm{L}}_{\mathrm{AP}} Q]\}$.
2. $\mathrm{LO}_{\mathrm{NLO}} = \{P \in \mathrm{NLO} \mid \exists Q \in \mathrm{AC}^0\mathrm{O}_{\mathrm{NLO}}\, [P \leq^{\mathrm{L}}_{\mathrm{EX}} Q]\}$.

In a given graph, a *path* of G is a sequence $(v_1, v_2, \ldots, v_k)$ of vertices satisfying that (v_i, v_{i+1}) is an edge for every index $i \in [k-1]$. A path is called *simple* if there are no repeated vertices in it. The *maximum vertex weight problem* (Max Vertex) takes a directed graph, a source $s \in V$, and a weight function $w : V \to \mathbb{N}^+$ and finds a path from s to a certain vertex $t \in V$ so that the weight of t is maximum. It follows from [7] that Max Vertex is $\leq^{\mathrm{L}}_{\mathrm{AP}}$-complete for $\mathrm{APXL}_{\mathrm{MaxNL}}$.

Proof Sketch of Proposition 1. We will show only (1). ($\subseteq$) Since Max Vertex is in $\mathrm{APXL}_{\mathrm{NLO}}$, take a constant $\gamma > 1$ and a log-space deterministic

Turing machine M that produces γ-approximate solutions for MAX VERTEX. Letting MAX VERTEX $= (I, SOL, m, \text{MAX})$, we modify it as follows and obtain a new problem, say, P_{max}. Instances of P_{max} are of the form (x, t_0), where $x \in I$ and $t_0 \in V$, satisfying the condition that (*) for every $v \in V$, $w(t_0) \leq w(v) \leq \gamma w(t_0)$. Consider an AC^0-circuit that outputs t_0 on input (x, t_0). Since $w(t_0) \leq m^*(x, t_0) \leq \gamma w(t_0)$, P_{max} must belong to $\mathrm{APXAC}^0_{\mathrm{NLO}}$.

Let $r \geq 1$ and define $f(x, r) = (x, t_0)$ and $g(x, y, r) = y$. Since t_0 can be obtained by running M on x, f is in FL. Note that the performance ratio $R_2(f(x, r), y)$ equals $R_1(x, g(x, y, r))$. Thus, MAX VERTEX APL-reduces to P. Moreover, because MAX VERTEX is $\leq^{\mathrm{L}}_{\mathrm{AP}}$-complete for $\mathrm{APXL}_{\mathrm{MaxNL}}$, we conclude that every maximization problem in $\mathrm{APXL}_{\mathrm{NLO}}$ is $\leq^{\mathrm{L}}_{\mathrm{AP}}$-reducible to P_{max}. The case of minimization is similar.

($\supseteq$) Let $P \in \mathrm{NLO}$ and $Q \in \mathrm{APXAC}^0_{\mathrm{NLO}}$ satisfying $P \leq^{\mathrm{L}}_{\mathrm{AP}} Q$. It is not difficult to prove that $Q \in \mathrm{APXAC}^0_{\mathrm{NLO}}$ implies $P \in \mathrm{APXL}_{\mathrm{NLO}}$. □

The complete problems presented in the proof of Proposition 1 does not seem to capture the essence of problems in $\mathrm{APXL}_{\mathrm{NLO}}$ as well as NLO. Therefore, in what follows, we intend to look into weaker notions of reducibilities. In particular, we want to limit our attention within $\leq^{\mathrm{NC}^1}_{\mathrm{AP}}$-complete and $\leq^{\mathrm{NC}^1}_{\mathrm{EX}}$-complete problems.

Let DSTCON denote the well-known *s-t connectivity problem* on directed graphs. Let us recall the *minimum path weight problem* (MIN PATH-WEIGHT) introduced in Section 1. Notice that, if we set a weight of every vertex of a given input graph to be 1, then MIN PATH-WEIGHT is equivalent to a problem of finding the shortest *s-t* path in the graph. We will prove that MIN PATH-WEIGHT is $\leq^{\mathrm{NC}^1}_{\mathrm{sAP}}$-complete for MinNL.

Proposition 2. MIN PATH-WEIGHT *is* $\leq^{\mathrm{NC}^1}_{\mathrm{sAP}}$*-complete for* MinNL.

Proof Sketch. For notational convenience, let MIN PATH-WEIGHT $= (I_0, SOL_0, m_0, \text{MIN})$. It is not difficult to show that MIN PATH-WEIGHT belongs to NLO. Next, we will show that every minimization problem in NLO is $\leq^{\mathrm{NC}^1}_{\mathrm{sAP}}$-reducible to MIN PATH-WEIGHT. Let $P = (I, SOL, m, \text{MIN})$ be any minimization problem in NLO. For m, we choose an appropriate log-space auxiliary Turing machine M with three tapes computing m. Recall that any solution candidate is written on M's auxiliary read-once tape. We define a *partial configuration* of M as a $\langle a, \sigma, b, \tau, c, u, d, \xi \rangle$, where an input tape-head scans σ at cell a, an auxiliary-tape head scans τ at cell b, u indicates the entire content of an $O(\log n)$-space work tape with its head scanning at cell c, and an output tape-head writes ξ in cell d, where all cell numbers are expressed in binary. The weight of this vertex is defined as ξ (expressed in binary). For convenience, we call this graph a *configuration graph* of M on input x. Let us define an instance of MIN PATH-WEIGHT as follows. Let $f(x, r)$ denote the configuration graph of M on input x. Let y be any path of the graph $f(x, r)$. Let $g(x, y, r)$ denote the content of the auxiliary tape that is reconstructed from labels attached to vertices along the path y. Clearly, f and g are in FNC^1. It is not difficult to

show that $m(f(x,r),y) = m_0(x,g(x,y,r))$. Therefore, MIN PATH-WEIGHT is $\leq_{\mathrm{sAP}}^{\mathrm{NC}^1}$-complete for MinNL. □

Under the assumption that L = NL, we can prove that the optimization problem MIN PATH-WEIGHT is $\leq_{\mathrm{sAP}}^{\mathrm{NC}^1}$-complete for NLO = MaxNL ∪ MinNL.

Lemma 2. *If* L = NL, *then* MIN PATH-WEIGHT *is* $\leq_{\mathrm{sAP}}^{\mathrm{NC}^1}$*-complete for* NLO.

Proof Sketch. By Proposition 2, it suffices to show that every maximization problem P_1 in NLO is sAPAC^0-reducible to a certain minimization problem P_2 in NLO (since AC$^0 \subseteq$ NC^1). Let $P_1 = (I_1, SOL_1, m_1, \mathrm{MAX})$ in NLO. We construct a minimization problem $P_2 = (I_2, SOL_2, m_2, \mathrm{MIN})$ in NLO as follows. Take an appropriate polynomial p satisfying that $b(x) = 2^{p(n)} \geq m_1^*(x)$ for every $x \in I$. Let $I_2 = I_1$ and $SOL_2 = SOL_1$. Moreover, for every $(x,y) \in I_2 \circ SOL_2$, let $m_2(x,y) = \lceil \frac{b(x)^2}{m_1(x,y)} \rceil$ if $m_1(x,y) > 0$; $b(x)^2$ otherwise. Here, we define $f(x,r) = x$ and $g(x,y,r) = y$. If $R_2(f(x,r),y) \leq r$ with $r \geq 1$, then $R_1(x, g(x,y,r))$ equals $\frac{m_1^*(x)}{m_1(x,y)}$, which is at most $\frac{b(x)^2}{m_2^*(x)+1} / \frac{b(x)^2}{m_2(x,y)} = \frac{m_2(x,y)}{m_2^*(x)} \leq \frac{rm_2^*(x)}{m_2^*(x)+1} \leq 1 + c(r-1)$, where $c = 1$, since $m_2(x,y) \leq rm_2^*(x)$.

To complete the proof, assuming that L = NL, we still need to prove that the measure function m_2 is in auxFL. Consider the following procedure: on input $x \in I$, guess a number e and a series of carry-on integers, check bit by bit whether $em_1(x,y) \leq b(x)^2$ and $(e+1)m_1(x,y) > b(x)^2$, check that all carry-on numbers are correct, and output e. Under the assumption of L = NL, this procedure can be implemented on a log-space auxiliary Turing machine. □

When we consider an undirected-graph version of MIN PATH-WEIGHT, denoted MIN UPATH-WEIGHT, it is log-space $2^{n^{O(1)}}$-approximable because, by the result of [6], using log space, we not only determine whether there exists a feasible solution for MIN UPATH-WEIGHT but also find at least one feasible solution if any. When all admissible input graphs of MIN UPATH-WEIGHT are restricted to be forests, we call the corresponding problem MIN FOREST-PATH-WEIGHT, where a *forest* is an acyclic undirected graph.

Proposition 3. MIN FOREST-PATH-WEIGHT *is* $\leq_{\mathrm{EX}}^{\mathrm{NC}^1}$*-complete for* $\mathrm{LO}_{\mathrm{NLO}}$.

Different from standard terminology, we will define a *mixed graph* $G = (V,E)$ to be induced from a directed graph (V_1, E_1) and an undirected graph (V_2, E_2) as $V = V_1 \times V_2$ and $E = \{((v_1,v_2),(v_1',v_2')) \mid (v_1,v_1') \in E_1, (v_2,v_2') \in E_2\}$.

The *minimum mixed path weight problem* (MIN MIX-PATH-WEIGHT) takes an instance of a mixed graph $G = (V,E)$ induced from (V_1,E_1) and (V_2,E_2), a source pair $(s_1,s_2) \in V$, and a weight function $w : V \to \mathbb{N} \times \mathbb{N}$ with two extra conditions: (i) (V_2,E_2) is a forest and (ii) $w_2(v_2) \leq w_1(v_1) \leq 2w_2(v_2)$ for every $(v_1,v_2) \in V$, where $w(v_1,v_2) = (w_1(v_1), w_2(v_2))$. The problem is to find a (mixed) path $\mathcal{S}$ of G starting at (s_1,s_2) and ending at (t_1,t_2) for which the partial path weight $\sum_{(v_1,v_2)\in\mathcal{S}} w_1(v_1)$ is minimum.

Proposition 4. MIN MIX-PATH-WEIGHT *is* $\leq_{\mathrm{sAP}}^{\mathrm{NC}^1}$*-complete for* $\mathrm{APXL}_{\mathrm{MinNL}}$.

Proof Sketch. It is not difficult to show that MIN MIX-PATH-WEIGHT is in $\mathrm{APXL}_{\mathrm{NLO}}$ (and thus $\mathrm{APXL}_{\mathrm{MinNL}}$). For simplicity, we set MIN MIX-PATH-WEIGHT $= (I_0, SOL_0, m_0, \mathrm{MIN})$ with measure function $m_0(x, \mathcal{S}) = \sum_{(v_1,v_2)\in\mathcal{S}} w_1(y_1)$.

Next, let $P = (I, SOL, m, \mathrm{MIN})$ be any minimization problem in $\mathrm{APXL}_{\mathrm{NLO}}$. Our goal is to show that P is $\leq_{\mathrm{AP}}^{\mathrm{NC}^1}$-reducible to MIN MIX-PATH-WEIGHT. Let M_1 be a log-space auxiliary Turing machine computing m and let M_2 be a log-space γ-approximation algorithm for P, where $\gamma > 1$ is a constant. This implies that (*) $m^*(x)/\gamma \leq m(x, M_2(x)) \leq m^*(x)$. Let M_3 compute $m(x, M_2(x))$ using log space. As in the proof of Proposition 2, we consider a pair of partial configurations of M_1 and M_3. Those pairs constitute a mixed graph. For each $i \in [3]$, let s_i be the initial configuration of M_i and let y_i be the final and accepting configuration of M_i. The weight of a path corresponds to the value of m. Given auxiliary input pair (z_1, z_3) of M_1 and M_3, let $h(z_1, z_3)$ denote the associated series of pairs of partial configurations of M_1 and M_3, respectively. Let us define $f(x, r)$ to be $\langle G, (s_1, s_2), (t_1, t_2), w\rangle$ and let $g(x, (y_1, y_3), r) = h^{-1}(y_1, y_3)$. The desired weight function $w(v_1, v_2) = (w_1(v_1), w_2(v_2))$ is defined as follows. Note that $m(x, y) = m_0(f(x, r), (y_1, y_3))$. If $\gamma \leq 2$, then Condition (*) implies $m_0^*(f(x, r))/2 \leq m_0(f(x, r), (y_1, y_3)) \leq m_0^*(f(x, r))$. Next, we assume that $\gamma > 2$. Fix $x \in I$. Let us define $\Delta = (\gamma - 2)m(x, M_2(x))$. We define $w_2(v_2)$ to be Δ plus the output value produced by M_1 that appears inside partial configuration v_2. Similarly, let $w_1(v_1)$ be Δ plus the value outputted by M_3 inside v_1. Note that w is computed from x using log space. Let $b(x) = m(x, M_2(x))$. The ratio $\frac{m_0(f(x,r),(y_1,y_3))}{m_0^*(f(x,r))}$ equals $\frac{b(x)+\Delta}{m^*(x)+\Delta} \leq \frac{b(x)+\Delta}{\gamma b(x)+\Delta} = \frac{1}{2}$ by (*), as requested. Therefore, $P \leq_{\mathrm{AP}}^{\mathrm{NC}^1}$ MIN-PATH-WEIGHT holds. □

3.2 Polynomially-Bounded Problems

For low-complexity optimization classes, polynomially-bounded optimization problems play a quite special role. Hereafter, we are focused on those problems.

Lemma 3. *1. Let P be a minimization (maximization, resp.) problem in* $\mathrm{APXL}_{\mathrm{NLO}} \cap \mathrm{PBO}$. *There exists a maximization (minimization, resp.) problem Q in* $\mathrm{APXL}_{\mathrm{NLO}} \cap \mathrm{PBO}$ *such that P is $\leq_{\mathrm{sAP}}^{\mathrm{AC}^0}$-reducible to Q.*

2. For any minimization (maximization, resp.) problem P in $\mathrm{NLO} \cap \mathrm{PBO}$, *there exists a maximization (minimization, resp.) problem Q in* $\mathrm{NLO} \cap \mathrm{PBO}$ *such that P is $\leq_{\mathrm{sAP}}^{\mathrm{AC}^0}$-reducible to Q.*

3. For any minimization (maximization, resp.) problem P in $\mathrm{LO}_{\mathrm{NLO}} \cap \mathrm{PBO}$, *there exists a maximization (minimization, resp.) problem Q in* $\mathrm{LO}_{\mathrm{NLO}} \cap \mathrm{PBO}$ *such that P is $\leq_{\mathrm{sAP}}^{\mathrm{AC}^0}$-reducible to Q.*

The *maximum bounded vertex weight problem* (MAX B-VERTEX) takes an undirected graph $G = (V, E)$, a source $s \in V$, and a weight function $w : V \to \mathbb{N}$ satisfying $w(v) \leq |V|$ for every $v \in V$, and finds a path of G starting at s and ending at a certain vertex t of the maximum non-zero weight.

Proposition 5. MAX B-VERTEX *is* $\leq_{\mathrm{EX}}^{\mathrm{NC}^1}$*-complete for* $\mathrm{LO}_{\mathrm{NLO}} \cap \mathrm{PBO}$.

The *maximum Boolean formula value problem* (MAX BFVP) takes a set of Boolean formulas and a Boolean assignment σ for variables in the formulas and finds a maximal set of satisfied formulas by σ. Note that MAX BFVP is known to be NC^1-complete.

Lemma 4. MAX BFVP *is* $\leq_{\mathrm{EX}}^{\mathrm{NC}^1}$*-complete for* $\mathrm{NC}^1\mathrm{O}_{\mathrm{NLO}} \cap \mathrm{PBO}$.

If we use $\leq_{\mathrm{AP}}^{\mathrm{L}}$-reductions instead of $\leq_{\mathrm{sAP}}^{\mathrm{NC}^1}$-reductions, then it is possible to prove that $\mathrm{APXL}_{\mathrm{NLO}}$ contains polynomially-bounded $\leq_{\mathrm{AP}}^{\mathrm{L}}$-complete problems.

Lemma 5. *There exists a polynomially-bounded optimization problem that is* $\leq_{\mathrm{AP}}^{\mathrm{L}}$*-complete for* $\mathrm{APXL}_{\mathrm{NLO}}$.

Proof Sketch. It is shown in [7] that $\mathrm{APXL}_{\mathrm{NLO}} \cap \mathrm{PBO}$ has an $\leq_{\mathrm{AP}}^{\mathrm{L}}$-complete maximization problem. To prove the lemma, we want to show that every minimization problem $P_1 = (I_1, SOL_1, m_1, \mathrm{MIN})$ in $\mathrm{APXL}_{\mathrm{NLO}}$ is $\leq_{\mathrm{AP}}^{\mathrm{L}}$-reducible to a certain maximization problem $P_2 = (I_2, SOL_2, m_2, \mathrm{MAX})$ in $\mathrm{APXL}_{\mathrm{NLO}} \cap \mathrm{PBO}$. Assume that, for an appropriate constant $\gamma > 1$, P_1 is γ-approximable by a log-space deterministic Turing machine M_1. Let $b(x) = m_1(x, M_1(x))$ for every $x \in I$. Note that $b(x)/\gamma \leq m_1^*(x) \leq b(x)$. For convenience, set $c = \gamma \log \gamma + \gamma - 1$.

Since the case where $1 + c(r-1) \geq \gamma$ is easy, we consider the other case where $1 + c(r-1) < \gamma$. For brevity, we set $\delta = 1 + c(r-1)$. Define $k = \lceil \log \gamma / \log \delta \rceil$. Note that $\delta^{k-1} \leq \gamma \leq \delta^k$. For convenience, we define $I_0 = \{((x, r, i) \mid x \in I, r \geq 1, 0 \leq i \leq k\}$ and $SOL_0(x, r, i) = \{y \in SOL_1(x) \mid m_1(x, y) \in (b(x)/\delta^{i+1}, b(x)/\delta^i]\}$. Let i_0 be the maximum integer i satisfying that $0 \leq i \leq k$ and $SOL_0(x, r, i) \neq \emptyset$. Note that $m_1^*(x) \in (b(x)/\delta^{i_0+1}, b(x)/\delta^{i_0}]$.

Let $I_2 = \{(x, r) \mid x \in I_1, r \geq 1\}$ and $SOL_2(x, r) = \{\langle y_0, y_1, \ldots, y_k\rangle \mid \exists i \in [0, k]_{\mathbb{Z}} \forall j \in [i+1, k]_{\mathbb{Z}} [y_i \in SOL_0(x, r, i) \wedge y_j \notin SOL_0(x, r, j)]\}$. Note that $I_2 \in \mathrm{L}$ and $I_2 \circ SOL_2 \in \mathrm{auxL}$. We set $m_2((x, r), y) = i + 1$ if $y = \langle y_0, \ldots, y_k\rangle$ and i is the maximum integer satisfying $y_i \in SOL_0(x, r, i)$. Note that $m_2 \in \mathrm{auxFL}$. Define $f(x, r) = (x, r)$ and $g((x, r), y, r) = y_i$ where $i = m_2((x, r), y)$. Take any $y \in SOL_2(x, r)$ for which $R_2(x, y) \leq r$. Since $m_2^*(x, r) = i_0 + 1$, it follows that $\frac{i_0+1}{r} \leq m_2((x, r), y) \leq i_0 + 1$. We then obtain $b(x)/\delta^{i_0+1} \leq m_1(x, y_i) \leq b(x)/\delta^{(i_0+1)/r}$. Thus, $R_1(x, g((x, r), y, r)) = R_1(x, y_i) = \frac{m_1(x,y_i)}{m_1^*(x)} \leq \frac{b(x)/\delta^{(i_0+1)/r}}{b(x)/\delta^{i_0+1}} = \delta^{(i_0+1)(1-1/r)}$. Since $\log z \geq \frac{z-1}{z}$ for all real numbers $z \in [1, 2]$, it follows that $k \leq \frac{\log \gamma}{\log \delta} + 1 \leq \frac{\delta \log \delta}{\delta - 1} + 1 = \frac{c}{\delta - 1}$. Hence, $r = \frac{\delta - 1}{c} + 1 \leq \frac{1}{k} + 1$. From this inequality, we obtain $(i_0 + 1)(1 - 1/r) \leq (k + 1)(1 - 1/r) = 1$. This implies $R_1(x, g((x, r), y, r)) \leq \delta = 1 + c(r - 1)$. Therefore, $P_1 \leq_{\mathrm{AP}}^{\mathrm{L}} P_2$ holds. □

4 Relations among Refined Optimization Classes

We will turn our attention to relationships among basic optimization problems introduced in Section 2. We start with claiming that two classes $\mathrm{APXP}_{\mathrm{NLO}}$ and $\mathrm{PO}_{\mathrm{NLO}}$ coincide with NLO.

Lemma 6. $\mathrm{APXP_{NLO}} = \mathrm{PO_{NLO}} = \mathrm{NLO}$.

Proof Sketch. Note that $\mathrm{PO_{NLO}} \subseteq \mathrm{APXP_{NLO}}$. First, we claim that $\mathrm{APXP_{NLO}} \subseteq \mathrm{NLO}$. By the definition of $\mathrm{APXP_{NLO}}$, all problems in $\mathrm{APXP_{NLO}}$ must be NLO problems, and hence they are in NLO. Next, we show that $\mathrm{NLO} \subseteq \mathrm{PO_{NLO}}$. Let $P = (I, SOL, m, goal)$ be any problem in NLO. We consider only the case of $goal = \textsc{max}$. We want to show that P also belongs to $\mathrm{PO_{NLO}}$. Let x be any instance in I. Consider the following algorithm on x. Define $D = \{(x, y) \in I \circ SOL \mid \exists z \in SOL(x)\,[z \geq y \wedge m(x, z) \geq m(x, y)]\}$, where $\geq$ is the lexicographic ordering. Note that $D \in \mathrm{NL} \subseteq \mathrm{P}$. By a binary search technique using D, we can find a maximal solution $y \in SOL^*(x)$ in polynomial time. Therefore, $\mathrm{NLO} \subseteq \mathrm{PO_{NLO}} \subseteq \mathrm{APXP_{NLO}} \subseteq \mathrm{NLO}$. □

Proposition 6. *1. [7]* $\mathrm{L} = \mathrm{NL}$ *iff* $\mathrm{LO_{NLO}} \cap \mathrm{PBO} = \mathrm{NLO} \cap \mathrm{PBO}$.
2. $\mathrm{NC^1} = \mathrm{L}$ *iff* $\mathrm{NC^1O_{NLO}} \cap \mathrm{PBO} = \mathrm{LO_{NLO}} \cap \mathrm{PBO}$.
3. $\mathrm{L} = \mathrm{P}$ *iff* $\mathrm{LO_{NPO}} \cap \mathrm{PBO} = \mathrm{PO_{NPO}} \cap \mathrm{PBO}$.

Proposition 7. *1. [7] If* $\mathrm{L} \neq \mathrm{NL}$, *then* $\mathrm{LO_{NLO}} \neq \mathrm{APXL_{NLO}} \neq \mathrm{NLO}$.
2. If $\mathrm{L} \neq \mathrm{P}$, *then* $\mathrm{PO_{NPO}} \not\subseteq \mathrm{APXL_{NPO}}$.
3. If $\mathrm{NC^1} \neq \mathrm{NL}$, *then* $\mathrm{NC^1O_{NLO}} \neq \mathrm{APXNC^1_{NLO}}$.

Proof Sketch. We will show only (3). Assume that $\mathrm{NC^1O_{NLO}} = \mathrm{APXNC^1_{NLO}}$. Consider DSTCON (on unweighted directed graphs), which is $\leq_m^{\mathrm{NC^1}}$-complete for NL. Taking a constant $\gamma > 1$, let us define a restricted version of Min Path-Weight, called Min Rest-Path$(\gamma) = (I, SOL, m, \textsc{min})$, as follows. An instance of Min Rest-Path is $w = \langle G, s, t, p_0\rangle$, where $G = (V, E)$ is a directed graph, and s, t are distinct vertices in G, p_0 is a special path from s to t. Let $len(p)$ denote the *length* of a path p. A solution of w is a path p from s to t satisfying $len(p_0)/\gamma \leq len(p) \leq len(p_0)$. We use the length of a path as the measure. It is not difficult to show that Min Rest-Path(γ) is in $\mathrm{APXAC^0_{NLO}}$, which is included in $\mathrm{APXNC^1_{NLO}}$.

By our assumption, Min Rest-Path$(\gamma) \in \mathrm{NC^1O_{NLO}}$ for any $\gamma > 1$. Now, we take $\gamma = 2$. Given an instance $\langle G, s, t\rangle$ of DSTCON, define $\langle G', s', t, p_0\rangle$ as follows. Let $n = |V|$ and let $G' = (V', E')$, where $V' = V \cup \{v_1, v_2, \ldots, v_n, s'\}$ and $E' = E \cup \{(s', v_1), (v_n, s), (s, w_1), (w_n, t)\} \cup \{(v_i, v_{i+1}), (w_i, w_{i+1}) \mid i \in [n-1]\}$. Moreover, let $p_0 = (s', v_1, v_2, \ldots, v_n, s, w_1, w_2, \ldots, w_n, t)$. If p is a path from s to t, let $p' = (s', v_1, \ldots, v_n, s) * p$, which is a concatenation of two paths. Since $len(p') = n + 1 + len(p)$, it follows that $n \leq len(p') \leq 2n$. Thus, an appropriate $\mathrm{NC^1}$-circuit computes the minimal path p by our assumption. If $len(p) < len(p_0)$, then we accept the input; otherwise, we reject the input.

Notice that $\langle G', s', t, p_0\rangle$ may be quite larger than $\langle G, s, t\rangle$ and, in general, we cannot produce $\langle G, s, t, p_0\rangle$ on a log-space work tape. However, we can avoid this pitfall as follows. Whenever a circuit needs information on vertices $\{v_1, \ldots, v_n, w_1, \ldots, w_n, s'\}$, the circuit automatically answer the question. This implies that DSTCON $\in \mathrm{NC^1}$. Since DSTCON is $\leq_m^{\mathrm{NC^1}}$-complete for NL, DSTCON $\in \mathrm{NC^1}$ implies $\mathrm{NC^1} = \mathrm{NL}$. □

Proposition 8. *1.* $\mathrm{NC}^1\mathrm{O}_{\mathrm{NLO}} \not\subseteq \mathrm{APXAC}^0_{\mathrm{NLO}}$.
2. $\mathrm{AC}^0\mathrm{O}_{\mathrm{NLO}} \neq \mathrm{APXAC}^0_{\mathrm{NLO}}$.

The *parity function* π is defined as $\pi(x_1, x_2, \ldots, x_n) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$, where each $x_i \in \{0,1\}$. Let $\pi^*(x_{11}, \ldots, x_{1n}, x_{21}, \ldots, x_{2n}, \ldots, x_{n1}, \ldots, x_{nn})$ be the n-bit string $\pi(x_{11}, \ldots, x_{1n})\pi(x_{21}, \ldots, x_{2n}) \cdots \pi(x_{n1}, \ldots, x_{nn})$. It is not difficult to show that π^* is in FNC^1 but not in FAC^0 because π resides in $\mathrm{NC}^1 - \mathrm{AC}^0$. Given $y \in \{0,1\}^+$, $rep(y)$ expresses *one plus* the natural number represented in binary as y.

Proof Sketch of Proposition 8. We will show only (1). Here, we consider the minimization problem Min m-Parity $= (I, SOL, m, \mathrm{MIN})$ defined as follows. Let $I = \bigcup_{n \in \mathbb{N}^+} \{0,1\}^{n^2}$ and $SOL(x) = \{y \in \{0,1\}^n \mid rep(y) \geq rep(\pi^*(x))\}$ for each $x \in I$ with $|x| = n^2$. Let $m(x, y) = rep(y)$. Clearly, I is in L, $I \circ SOL$ is in auxL, and m is in FNC^1. Hence, Min m-Parity $\in$ NLO. Since $SOL^*(x) = \{\pi^*(x)\}$ for every $x \in I$, it follows from $\pi^* \in \mathrm{FNC}^1$ that Min m-Parity is NC^1-solvable.

Next, we will prove that Min m-Parity $\notin \mathrm{APXAC}^0_{\mathrm{NLO}}$. Assume otherwise. There exists a uniform family $\{C_n\}_{n \in \mathbb{N}^+}$ of AC^0-circuits such that, for every $x \in I$, $C_{|x|}(x)$ computes a string y in $SOL(x)$ such that $(1/\gamma)rep(y) \leq rep(\pi^*(x)) \leq rep(y)$. Take any number n satisfying $2^n > \gamma$ and any string $x \in \{0,1\}^n$. Let us consider $\pi^*(x^n)$. Define $C_{n^2}(x^n) = y$. If $\pi(x) = 1$, then we have $2^{n+1} \leq rep(y) \leq \gamma \cdot 2^{n+1}$ because of $rep(\pi^*(x^n)) = 2^{n+1}$. Since $|y| = n$, it must hold that $rep(y) = 2^{n+1}$; that is, $y = 1^n$. However, if $\pi(x) = 0$, then we obtain $1 \leq rep(y) \leq \gamma$ since $rep(\pi^*(x^n)) = 1$. Since $\gamma < 2^n$, y has the form $0z$. Hence, $\pi(x)$ equals the first bit of y. This gives an AC^0-circuit that computes π. This is a contradiction against the fact that π is not in AC^0. Therefore, Min m-Parity does not belong to $\mathrm{APXAC}^0_{\mathrm{NLO}}$. □

References

1. Àlvarez, C., Jenner, B.: A very hard log-space counting class. Theoret. Comput. Sci. 107, 3–30 (1993)
2. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer (2003)
3. Gabow, H.N.: A matroid approach to finding edge connectivity and packing arborescences. In: STOC 1991, pp. 112–122. ACM Press (1991)
4. Goldschlager, L.M., Shaw, R.A., Staples, J.: The maximum flow problem is log space complete for P. Theoret. Comput. Sci. 21, 105–111 (1982)
5. Karger, D.R.: Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. In: SODA 1993, pp. 21–30 (1993)
6. Reingold, O.: Undirected connectivity in log-space. J. ACM 55, article 17 (2008)
7. Tantau, T.: Logspace optimisation problems and their approximation properties. Theory Comput. Syst. 41, 327–350 (2007)