A Weighted Linear Matroid Parity Algorithm

Satoru Iwata
University of Tokyo
Tokyo 113-8656, Japan
iwata@mist.i.u-tokyo.ac.jp

Yusuke Kobayashi
University of Tsukuba
Tsukuba, Ibaraki 305-8573, Japan
kobayashi@sk.tsukuba.ac.jp

ABSTRACT
The matroid parity (or matroid matching) problem, introduced as a common generalization of matching and matroid intersection problems, is so general that it requires an exponential number of oracle calls. Lovász (1980) showed that this problem admits a min-max formula and a polynomial algorithm for linearly represented matroids. Since then, efficient algorithms have been developed for the linear matroid parity problem.

In this paper, we present a combinatorial, deterministic, polynomial-time algorithm for the weighted linear matroid parity problem. The algorithm builds on a polynomial matrix formulation using Pfaffian and adopts a primal-dual approach based on the augmenting path algorithm of Gabow and Stallmann (1986) for the unweighted problem.

CCS CONCEPTS
• Mathematics of computing → Matroids and greedoids; Combinatorial optimization; Combinatorial algorithms;

KEYWORDS
Linear matroid parity, matching, polynomial-time algorithm, Pfaffian, primal-dual approach

1 INTRODUCTION
The matroid parity problem [17] (also known as the matroid matching problem [16] or the matroid matching problem [18]) was introduced as a common generalization of matching and matroid intersection problems. In the worst case, it requires an exponential number of independence oracle calls [15, 20]. Nevertheless, Lovász [18, 20, 21] showed that the problem admits a min-max theorem for linear matroids and presented a polynomial algorithm that is applicable if the matroid in question is represented by a matrix.

Since then, efficient combinatorial algorithms have been developed for this linear matroid parity problem [8, 27, 28]. Gabow and Stallmann [8] developed an augmenting path algorithm with the aid of a linear algebraic trick, which was later extended to the linear delta-matroid parity problem [10]. Orlin and Vande Vate [28] provided an algorithm that solves this problem by repeatedly solving matroid intersection problems coming from the min-max theorem. Later, Orlin [27] improved the running time bound of this algorithm. The current best deterministic running time bound due to [8, 27] is \(O(nm^ω)\), where \(n\) is the cardinality of the ground set, \(m\) is the rank of the linear matroid, and \(ω\) is the matrix multiplication exponent, which is at most 2.38. These combinatorial algorithms, however, tend to be complicated.

An alternative approach that leads to simpler randomized algorithms is based on an algebraic method. This is originated by Lovász [19], who formulated the linear matroid parity problem as rank computation of a skew-symmetric matrix that contains independent parameters. Substituting randomly generated numbers to these parameters enables us to compute the optimal value with high probability. A straightforward adaptation of this approach requires iterations to find an optimal solution. Cheung, Lau, and Leung [3] have improved this algorithm to run in \(O(nm^{ω−1})\) time, extending the techniques of Harvey [12] developed for matching and matroid intersection.

While matching and matroid intersection algorithms have been successfully extended to their weighted version, no polynomial algorithms have been known for the weighted linear matroid parity problem for more than three decades. Camerini, Galbiati, and Maffioli [2] developed a random pseudopolynomial algorithm for the weighted linear matroid parity problem by introducing a polynomial matrix formulation that extends the matrix formulation of Lovász [19]. This algorithm was later improved by Cheung, Lau, and Leung [3]. The resulting complexity, however, remained pseudopolynomial. Tong, Lawler, and Vazirani [33] observed that the weighted matroid parity problem on gammoids can be solved in polynomial time by reduction to the weighted matching problem. As a relaxation of the matroid matching polytope, Vande Vate [34] introduced the fractional matroid matching polytope. Gijswijt and Pap [11] devised a polynomial algorithm for optimizing linear functions over this polytope. The polytope was shown to be half-integral, and the algorithm does not necessarily yield an integral solution.

In this paper, we give a combinatorial, deterministic, polynomial-time algorithm for the weighted linear matroid parity problem. To do so, we combine algebraic approach and augmenting path technique together with the use of node potentials. The algorithm builds on a polynomial matrix formulation, which naturally extends the one discussed in [9] for the unweighted problem. The algorithm employs a modification of the augmenting path search procedure for the unweighted problem by Gabow and Stallmann [8]. It adopts
The correctness proof for the optimality is based on the idea of combinatorial relaxation for polynomial matrices due to Murota [25]. The algorithm is shown to require \(O(n^2 m)\) arithmetic operations. This leads to a strongly polynomial algorithm for linear matroids represented over a finite field. For linear matroids represented over the rational field, one can exploit our algorithm to solve the problem in polynomial time.

Independently of the present work, Gyula Pap has obtained another combinatorial, deterministic, polynomial-time algorithm for the weighted linear matroid parity problem based on a different approach.

The matroid matching theory of Lovász [21] in fact deals with more general class of matroids that enjoy the double circuit property. Dress and Lovász [6] showed that algebraic matroids satisfy this property. Subsequently, Hochstädtler and Kern [13] showed the same phenomenon for pseudomodular matroids. The min-max theorem follows for this class of matroids. To design a polynomial algorithm, however, one has to establish how to represent those matroids in a compact manner. Extending this approach to the weighted problem is left for possible future investigation.

The algorithm is shown to require \(O(n^2 m)\) arithmetic operations. This leads to a strongly polynomial algorithm for linear matroids represented over a finite field. For linear matroids represented over the rational field, one can exploit our algorithm to solve the problem in polynomial time.

The linear matroid parity problem finds various applications: structural solvability analysis of passive electric networks [24], pinning down planar skeleton structures [22], and maximum genus cellular embedding of graphs [7]. We describe below two interesting applications of the weighted matroid parity problem in combinatorial optimization.

A \(T\)-path in a graph is a path between two distinct vertices in the terminal set \(T\). Mader [23] showed a min-max characterization of the maximum number of openly disjoint \(T\)-paths. The problem can be equivalently formulated in terms of \(S\)-paths, where \(S\) is a partition of \(T\) and an \(S\)-path is a \(T\)-path between two different components of \(S\). Lovász [21] formulated the problem as a matroid matching problem and showed that one can find a maximum number of disjoint \(S\)-paths in polynomial time. Schrijver [31] has described a more direct reduction to the linear matroid parity problem.

The disjoint \(S\)-paths problem has been extended to path packing problems in group-labeled graphs [4, 5, 29]. Tanigawa and Yamaguchi [32] have shown that these problems also reduce to the matroid matching problem with double circuit property. Yamaguchi [35] clarifies a characterization of the groups for which those problems reduce to the linear matroid parity problem.

As a weighted version of the disjoint \(S\)-paths problem, it is quite natural to think of finding disjoint \(S\)-paths of minimum total length. It is not immediately clear that this problem reduces to the weighted linear matroid parity problem. A recent paper of Yamaguchi [36] clarifies that this is indeed the case. He also shows that the reduction results on the path packing problems on group-labeled graphs also extend to the weighted version.

The weighted linear matroid parity is also useful in the design of approximation algorithms. Prömel and Steger [30] provided an approximation algorithm for the Steiner tree problem. Given an instance of the Steiner tree problem, construct a hypergraph on the terminal set such that each hyperedge corresponds to a terminal subset of cardinality at most three and regard the shortest length of a Steiner tree for the terminal subset as the cost of the hyperedge. The problem of finding a minimum cost spanning hypertree in the resulting hypergraph can be converted to the problem of finding minimum spanning tree in a 3-uniform hypergraph, which is a special case of the weighted parity problem for graphic matroids. The minimum spanning hypertree thus obtained costs at most \(5/3\) of the optimal value of the original Steiner tree problem, and one can construct a Steiner tree from the spanning hypertree without increasing the cost. Thus they gave a \(5/3\)-approximation algorithm for the Steiner tree problem via weighted linear matroid parity. This is a very interesting approach that suggests further use of weighted linear matroid parity in the design of approximation algorithms, even though the performance ratio is larger than the current best one for the Steiner tree problem [1].

2 THE MINIMUM-WEIGHT PARITY BASE PROBLEM

Let \(A\) be a matrix of row-full rank over an arbitrary field \(K\) with row set \(U\) and column set \(V\). Assume that both \(m = |U|\) and \(n = |V|\) are even. The column set \(V\) is partitioned into pairs, called lines. Each \(v \in V\) has its mate \(\bar{v}\) such that \((v, \bar{v})\) is a line. We denote by \(L\) the set of lines, and suppose that each line \(\ell \in L\) has a weight \(w_\ell \in \mathbb{R}\).

The linear dependence of the column vectors naturally defines a matroid \(M(A)\) on \(V\). Let \(B\) denote its base family. A base \(B \in B\) is called a parity base if it consists of lines. As a weighted version of the linear matroid parity problem, we will consider the problem of finding a parity base of minimum weight, where the weight of a parity base is the sum of the weights of lines in it. We denote the optimal value by \(\zeta(A, L, w)\). This problem generalizes finding a minimum-weight perfect matching in graphs and a minimum-weight common base of a pair of linear matroids on the same ground set.

As another weighted version of the matroid parity problem, one can think of finding a matching (independent parity set) of maximum weight. This problem can be easily reduced to the minimum-weight parity base problem.

Associated with the minimum-weight parity base problem, we consider a skew-symmetric polynomial matrix \(\Phi_A(\theta)\) in variable \(\theta\) defined by

\[
\Phi_A(\theta) = \begin{pmatrix} O & A \\ -A^T & D(\theta) \end{pmatrix},
\]

where \(D(\theta)\) is a block-diagonal matrix in which each block is a \(2 \times 2\) skew-symmetric polynomial matrix

\[
D_\ell(\theta) = \begin{pmatrix} 0 & -\tau_\ell \theta w_\ell \\ \tau_\ell \theta w_\ell & 0 \end{pmatrix}
\]

corresponding to a line \(\ell \in L\). Assume that the coefficients \(\tau_\ell\) are independent parameters (or indeterminates).

For a skew-symmetric matrix \(\Phi\) whose rows and columns are indexed by \(W\), the support graph of \(\Phi\) is the graph \(\Gamma = (W, E)\) with edge set \(E = \{(u, v) \mid \Phi_{uv} \neq 0\}\). We denote by \(\text{Pf}\Phi\) the Pfaffian of \(\Phi\), which is defined as follows:

\[
\text{Pf}\Phi = \sum_M \sigma_M \prod_{(u, v) \in M} \Phi_{uv}.
\]
where the sum is taken over all perfect matchings \( M \) in \( \Gamma \) and \( \sigma_M \) takes \( \pm 1 \) in a suitable manner, see [22]. It is well-known that \( \det \Phi = (\det \Phi)^2 \) and \( \det (\Phi^T \Phi) = \Phi \cdot \det S \) for any square matrix \( S \).

We have the following lemma that associates the optimal value of the minimum-weight parity base problem with \( \Phi \Phi_A(\theta) \).

**Lemma 2.1.** The optimal value of the minimum-weight parity base problem is given by

\[
\zeta(A, L, w) = \sum_{\ell \in L} w_\ell - \deg_\theta \Phi \Phi_A(\theta).
\]

In particular, if \( \Phi \Phi_A(\theta) = 0 \) (i.e., \( \deg_\theta \Phi \Phi_A(\theta) = -\infty \)), then there is no parity base.

**Proof.** We split \( \Phi \Phi_A(\theta) \) into \( \Psi_A \) and \( \Delta(\theta) \) such that \( \Phi \Phi_A(\theta) = \Psi_A + \Delta(\theta) \).

\[
\Psi_A = \begin{pmatrix} O & A \\ -A^T & O \end{pmatrix}, \quad \Delta(\theta) = \begin{pmatrix} O & O \\ O & D(\theta) \end{pmatrix}.
\]

The row and column sets of these skew-symmetric matrices are indexed by \( W := U \cup V \). By [26, Lemma 7.3.20], we have

\[
\Phi \Phi_A(\theta) = \sum_{X \subseteq W} \pm \Phi \Psi_A[W \setminus X] \cdot \Phi \Delta(\theta)[X],
\]

where each sign is determined by the choice of \( X \), \( \Delta(\theta)[X] \) is the principal submatrix of \( \Delta(\theta) \) whose rows and columns are both indexed by \( X \), and \( \Psi_A[W \setminus X] \) is defined in a similar way. One can see that \( \Phi \Phi_A(\theta) = 0 \) if and only if \( X \subseteq V \) or, equivalently \( B := V \setminus X \) is a union of lines. One can also see for \( X \subseteq V \) that \( \Phi \Psi_A[W \setminus X] = 0 \) if and only if \( A[U, V \setminus X] \) is nonsingular, which means that \( B \) is a base of \( M(A) \). Thus, we have

\[
\Phi \Phi_A(\theta) = \sum_B \pm \Phi \Psi_A[U \cup B] \cdot \Phi \Delta(\theta)[V \setminus B],
\]

where the sum is taken over all parity bases \( B \). Note that no term is canceled out in the summation, because each term contains a distinct set of independent parameters. For a parity base \( B \), we have

\[
\deg_\theta(\Phi \Psi_A[U \cup B] \cdot \Phi \Delta(\theta)[V \setminus B]) = \sum_{\ell \subseteq V \setminus B} w_\ell = \sum_{\ell \subseteq B} w_\ell - \sum_{\ell \subseteq L} w_\ell,
\]

which implies that the minimum weight of a parity base is

\[
\sum_{\ell \in L} w_\ell - \deg_\theta \Phi \Phi_A(\theta).
\]

\[\square\]

Note that Lemma 2.1 does not immediately lead to a (randomized) polynomial-time algorithm for the minimum weight parity base problem. This is because computing the degree of the Pfaffian of a skew-symmetric polynomial matrix is not so easy. Indeed, the algorithms in [2, 3] for the weighted linear matroid parity problem compute the degree of the Pfaffian of another skew-symmetric polynomial matrix, which results in pseudopolynomial complexity.

### 3 ALGORITHM OUTLINE

In this section, we describe the outline of our algorithm for solving the minimum-weight parity base problem.

We regard the column set \( V \) as a vertex set. The algorithm works on a vertex set \( V^* \supseteq V \) that includes some new vertices generated during the execution. The algorithm keeps a nested (laminar) collection \( \Lambda = \{H_1, \ldots, H_{|\Lambda|}\} \) of vertex subsets of \( V^* \) such that \( H_i \cap V \) is a set of lines for each \( i \). The indices satisfy that, for any two members \( H_i, H_j \in \Lambda \) with \( i < j \), either \( H_i \cap H_j = \emptyset \) or \( H_i \subsetneq H_j \) holds. Each member of \( \Lambda \) is called a blossom. The algorithm maintains a potential \( p : V^* \to \mathbb{R} \) and a nonnegative variable \( q : \Lambda \to \mathbb{R}_+ \), which are collectively called dual variables. It also keeps a subset \( B^* \subseteq V^* \) such that \( B := B^* \cap V \in \mathcal{E} \).

We note that although \( p \) and \( q \) are called dual variables, they do not correspond to dual variables of an LP-relaxation of the minimum-weight parity base problem. Indeed, this paper presents neither an LP-formulation nor a min-max formula for the minimum-weight parity base problem, explicitly. We will show instead that one can obtain a parity base \( B \) that admits feasible dual variables \( p \) and \( q \), which provide a certificate for the optimality of \( B \).

The algorithm starts with splitting the weight \( w_\ell \) into \( p(\ell) \) and \( p(\bar{\ell}) \) for each line \( \ell = \{v, \bar{v}\} \in L \), i.e., \( p(\ell) + p(\bar{\ell}) = w_\ell \). Then it executes the greedy algorithm for finding a base \( B \in \mathcal{E} \) with minimum value of \( p(B) = \sum_{u \in B} p(u) \). If \( B \) is a parity base, then \( B \) is obviously a minimum-weight parity base. Otherwise, there exists a line \( \ell = \{v, \bar{v}\} \) in which exactly one of its two vertices belongs to \( B \). Such a line is called a source line and each vertex in a source line is called a source vertex. A line that is not a source line is called a normal line.

The algorithm initializes \( \Lambda := \emptyset \) and proceeds iterations of primal and dual updates, keeping dual feasibility. In each iteration, the algorithm applies the breadth-first search to find an augmenting path. In the meantime, the algorithm sometimes detects a new blossom and adds it to \( \Lambda \). If an augmenting path \( P \) is found, the algorithm updates \( B \) along \( P \). This will reduce the number of source lines by two. If the search procedure terminates without finding an augmenting path, the algorithm updates the dual variables to create new tight edges. The algorithm repeats this process until \( B \) becomes a parity base. Then \( B \) is a minimum-weight parity base.

The rest of this paper is organized as follows. In Section 4, we introduce new notions attached to blossoms. The feasibility of the dual variables is defined in Section 5. In Section 6, we show that a parity base that admits feasible dual variables attains the minimum weight. In Section 7, we describe a search procedure for an augmenting path. The validity of the procedure is shown in Section 8. In Section 9, we describe how to update the dual variables when the search procedure terminates without finding an augmenting path. If the search procedure succeeds in finding an augmenting path \( P \), the algorithm updates the base \( B \) along \( P \). The details of this process is presented in Section 10. Finally, in Section 11, we describe the entire algorithm and analyze its running time. Due to the space limitation, Sections 8 and 10 are omitted in this proceedings version.
4 BLOSSOMS

In this section, we introduce buds and tips attached to blossoms and construct auxiliary matrices that will be used in the definition of dual feasibility.

Each blossom contains at most one source line, and a blossom that contains a source line is called a source blossom. A blossom with no source line is called a normal blossom. Let $\Lambda_v$ and $\Lambda_a$ denote the sets of source blossoms and normal blossoms, respectively. Each normal blossom $H_i \in \Lambda_a$ contains mutually disjoint vertices $b_i$, $t_i$, and $t_i$ outside $V$, where $b_i$, $t_i$, and $t_i$ are called the bud of $H_i$, the tip of $H_i$, and the mate of $t_i$, respectively. The vertex set $V^*$ is defined by $V^* := V \cup \{b_i, t_i, t_i \mid H_i \in \Lambda_a\}$. For every $i$, if $H_i \in \Lambda_a$, we have $b_i, t_i, t_i \in H_i$ if and only if $H_i \subseteq H_i$ (see Fig. 1). Although $t_i$ is called the mate of $t_i$, we call $(t_i, t_i)$ a dummy line instead of a line. If $H_i \in \Lambda_a$, we regard $\{b_i\}, \{t_i\}$, and $\{t_i\}$ as $0$. The algorithm keeps a subset $B^* \subseteq V$ such that $B := B^* \cap V \in \mathcal{B}$ and $|B^* \cap \{b_i, t_i, t_i\}| = |B \cap \{b_i, t_i, t_i\}| = 1$ for each $i$ with $H_i \in \Lambda_a$. It also keeps $H_i \cap V \neq H_i \cap V$ for distinct $H_i, H_j \in \Lambda$. This implies that $|\Lambda| = O(n)$, where $n = |V|$, and hence $|V^*| = O(n)$.

Figure 1: Illustration of blossoms. Black nodes are in $B^*$ and white nodes are in $V^* \setminus B^*$.

Recall that $U$ is the row set of $A$. The fundamental circuit matrix $C$ with respect to a base $B$ is a matrix with row set $B$ and column set $V \setminus B$ obtained by $C = [A(U, B)]^{-1}A(U, V \setminus B)$. In other words, $[C]$ is obtained from $A$ by identifying $B$ and $U$, applying row transformations, and changing the ordering of columns. We keep a matrix $C^*$ whose row and column sets are $B^*$ and $V^* \setminus B^*$, respectively, such that the restriction of $C^*$ to $V^*$ is the fundamental circuit matrix $C$ with respect to $B$, that is, $C = C^*[V \setminus B^*, V \setminus B^*]$. If the row and column sets of $C^*$ are clear, for a vertex set $X \subseteq V^*$, we denote $C^*[X \setminus B^*, X \setminus B^*]$ by $C^*[X]$. For each $i$ with $H_i \in \Lambda_a$, the matrix $C^*$ satisfies the following properties.

\begin{itemize}
  \item If $b_i, t_i \in B^*$ and $t_i \in V^* \setminus B^*$, then $C^*_{b_i t_i} = 0$, $C^*_{t_i t_i} \neq 0$, $C^*_{b_i v} = 0$ for any $v \in (V^* \setminus B^*) \setminus H_i$, and $C^*_{v t_i} = 0$ for any $v \in (V^* \setminus B^*) \setminus \{t_i\}$ (see Fig. 2).
  \item If $b_i, t_i \in V^* \setminus B^*$ and $t_i \in B^*$, then $C^*_{b_i t_i} = 0$, $C^*_{t_i t_i} = 0$, $C^*_{b_i v} = 0$ for any $v \in (V^* \setminus B^*) \setminus H_i$, and $C^*_{u t_i} = 0$ for any $u \in B^* \setminus \{t_i\}$.
\end{itemize}

Figure 2: Illustration of (BT). Real lines represent nonzero entries of $C^*$.

We henceforth denote by $\lambda$ the current number of blossoms, i.e., $\lambda := |\Lambda|$. For $i = 0, 1, \ldots, \lambda$, we recursively define a matrix $C^i$ with row set $B^*$ and column set $V^* \setminus B^*$ so that $C^i$ is obtained from $C^{i-1}$ by adding a constant multiplication of the row (resp. column) corresponding to $t_i$ to other rows (resp. columns) as follows. Set $C^0 := C_*$. For $i \geq 1$, if $H_i \in \Lambda_a$, then define $C^i := C^{i-1}$. Otherwise, define $C^i$ as follows.

\begin{itemize}
  \item If $b_i \in B^*$ and $t_i \in V^* \setminus B^*$, then $C^i$ is defined to be the matrix obtained from $C^{i-1}$ by a column transformation eliminating $C^{i-1}_{b_i t_i}$ for every $v \in (V^* \setminus B^*) \setminus \{t_i\}$. That is,
    \[ C^i_{u v} = \begin{cases} 
      C^{i-1}_{u v} - C^{i-1}_{u b_i} C^{i-1}_{b_i t_i} & \text{if } v \neq t_i, \\
      C^{i-1}_{u v} & \text{if } v = t_i.
    \end{cases} \]
  \item If $b_i \in V^* \setminus B^*$ and $t_i \in B^*$, then $C^i$ is defined to be the matrix obtained from $C^{i-1}$ by a row transformation eliminating $C^{i-1}_{u b_i}$ for every $u \in B^* \setminus \{t_i\}$. That is,
    \[ C^i_{u v} = \begin{cases} 
      C^{i-1}_{u v} - C^{i-1}_{u t_i} C^{i-1}_{t_i b_i} & \text{if } u \neq t_i, \\
      C^{i-1}_{u v} & \text{if } u = t_i.
    \end{cases} \]
\end{itemize}

In the definition of $C^i$, we use the fact that $C^i_{b_i t_i} \neq 0$ or $C^i_{t_i b_i} \neq 0$, which is guaranteed by the following lemma.

**Lemma 4.1.** For any $j \in \{0, 1, \ldots, \lambda\}$ and $i \in \{1, \ldots, \lambda\}$ with $H_i \in \Lambda_a$, the following statements hold.

1. If $b_i, t_i \in B^*$ and $t_i \in V^* \setminus B^*$, then we have the following.
   \begin{align*}
   & (1-1) \quad C^i_{b_i v} = 0 \quad \text{for any } v \in (V^* \setminus B^*) \setminus H_i \quad \text{and} \quad C^i_{b_i t_i} = C^i_{b_i t_i} \neq 0, \\
   & (1-2) \quad C^i_{t_i v} = 0 \quad \text{for any } v \in (V^* \setminus B^*) \setminus H_i.
   \end{align*}

2. Suppose that a vertex $u \in B^*$ satisfies that $C^i_{u v} = 0$ for any $v \in (V^* \setminus B^*) \setminus H_i$. If $j \geq i$, then $C^j_{u v} = C^i_{u v}$ for any $v \in (V^* \setminus B^*) \setminus H_i$.

3. If $b_i, t_i \in V^* \setminus B^*$ and $t_i \in B^*$, then we have the following.
   \begin{align*}
   & (2-1) \quad C^i_{u t_i} = 0 \quad \text{for any } u \in B^* \setminus H_i \quad \text{and} \quad C^i_{t_i b_i} = C^i_{t_i b_i} \neq 0, \\
   & (2-2) \quad C^i_{u b_i} = 0 \quad \text{for any } u \in B^* \setminus H_i.
   \end{align*}

The proof is given in the full version [14].

5 DUAL FEASIBILITY

In this section, we define feasibility of the dual variables and show their properties. Our algorithm is designed on the minimum-weight parity base problem which is designed so that it keeps the dual feasibility.

Recall that a potential $p : V^* \to \mathbb{R}$, and a nonnegative variable $q : \Lambda \to \mathbb{R}_+$ are called dual variables. A blossom $H_i$ is said to be positive if $g(H_i) > 0$. For distinct vertices $u, v \in V^*$ and for $H_i \in \Lambda$, we say that a pair $(u, v)$ crosses $H_i$ if $[u, v] \cap H_i = 1$. For distinct $u, v \in V^*$, we denote by $iuv$ the set of indices $i \in \{1, \ldots, \lambda\}$ such that $(u, v)$ crosses $H_i$. The maximum element of $iuv$ is denoted by $iuv$. We also denote by $juv$ the set of indices $i \in \{1, \ldots, \lambda\}$ such that $t_j \in \{u, v\}$. We introduce the set $F_A$ of ordered vertex pairs.
defined by

\[ F_A := \{(u, v) \mid u \in B^+, v \in V^+ \setminus B^+, C^w_{uv} \neq 0\}. \]

Note that \( F_A \) is closely related to the nonzero entries in \( C^\Lambda \) as we will see later in Observation 7.1. For \( u, v \in V^+ \), we define

\[ Q_{uv} := \sum_{i \in I_{uv}} q(H_i) = \sum_{i \in I_{uv}} q(H_i). \]

The dual variables are called feasible with respect to \( C^\Lambda \) and \( \Lambda \) if they satisfy the following.

(DF1) \( p(v) + p(\tilde{v}) = w_{\ell} \) for every line \( \ell = \{v, \tilde{v}\} \in L \).
(DF2) \( p(v) - p(u) \geq Q_{uv} \) for every \( (u, v) \in F_A \).
(DF3) \( p(b_j) = p(t_j) \) for every \( H_j \in \Lambda_n \).

If no confusion may arise, we omit \( C^\Lambda \) and \( \Lambda \) when we discuss dual feasibility.

Note that if \( p \) satisfies (DF1), \( \Lambda = \emptyset \), and \( B \in \mathcal{B} \) minimizes \( p(B) = \sum_{u \in B} p(u) \) in \( B \), then \( p \) and \( q \) are feasible. This ensures that the initial setting of the algorithm satisfies the dual feasibility.

We now show some properties of feasible dual variables.

**Lemma 5.1.** For distinct vertices \( u \in B^+ \) and \( v \in V^+ \setminus B^+ \), we have \( (u, v) \in F_A \) if and only if \( C[X] \) is nonsingular, where \( X := \{u, v\} \cup \bigcup \{(b_i, t_i) \mid i \in I_{uv} \setminus J_{uv}, H_i \in \Lambda_n\} \).

**Proof.** By (1-3) and (2-3) of Lemma 4.1, if \( b_i \in B^+ \) for \( i \in I_{uv} \setminus J_{uv} \), then \( C_{u'b_i}^{w} = 0 \) for any \( u' \in (V^+ \setminus B^+) \setminus \{t_i\} \) and \( C_{u't_i}^{w} \neq 0 \), and if \( b_i \in V^+ \setminus B^+ \) for \( i \in I_{uv} \setminus J_{uv} \), then \( C_{u'b_i}^{w} = 0 \) for any \( u' \in B^+ \setminus \{t_i\} \) and \( C_{u't_i}^{w} \neq 0 \). This implies that \( C_{uv}^{w} \neq 0 \) is equivalent to that \( C^[w][X] \) is nonsingular.

For any \( j \in \{1, \ldots, \mu\} \), either \( X \cap H_j = \emptyset \) or \( t_j \in H_j \) holds. By (1-1) and (2-1) of Lemma 4.1, this shows that either \( C^[w][X] = C^[w]^{-1}[X] \) or \( C^[w][X] \) is obtained from \( C^[w]^{-1}[X] \) by applying elementary transformations. Therefore, \( C^[w][X] \) is obtained from \( C^[w][X] \) by applying elementary transformations, and hence the nonsingularity of \( C^[w][X] \) is equivalent to that of \( C^[w][X] \).

When we are given a set of blossoms \( \Lambda \), there may be more than one way of indexing the blossoms so that for any two members \( H_i, H_j \in \Lambda \) with \( i < j \), either \( H_i \cap H_j = \emptyset \) or \( H_i \subseteq H_j \) holds. Lemma 5.1 guarantees that this flexibility does not affect the definition of the dual feasibility. Thus, we can renumber the indices of the blossoms if necessary.

The following lemma guarantees that we can remove (or add) a blossom \( H \) with \( q(H) = 0 \) from (or to) \( \Lambda \). The proof is given in the full version [14].

**Lemma 5.2.** Suppose that \( p : V^+ \rightarrow \mathbb{R} \) and \( q : \Lambda \rightarrow \mathbb{R}_+ \) are dual variables, and let \( i \in \{1, 2, \ldots, \lambda\} \) be an index such that \( q(H_i) = 0 \). Suppose that \( p(b_i) = p(t_i) = p(t_j) \) if \( H_i \in \Lambda_n \). Let \( q' \) be the restriction of \( q \) to \( \Lambda' := \Lambda \setminus \{H_i\} \). Then \( p \) and \( q' \) are feasible with respect to \( \Lambda \) if and only if \( p \) and \( q \) are feasible with respect to \( \Lambda' \). Here, we do not remove \( \{b_i, t_i, t_j\} \) from \( V^+ \) even when we consider dual feasibility with respect to \( \Lambda' \).

The next lemma shows that \( p(v) - p(u) \geq Q_{uv} \) holds if \( u \) and \( v \) satisfy a certain condition. The proof is given in the full version [14].

**Lemma 5.3.** Let \( p \) and \( q \) be feasible dual variables and let \( k \in \{0, 1, \ldots, \lambda\} \). For any \( u \in B^+ \) and \( v \in V^+ \setminus B^+ \) with \( i_{uv} \leq k \) and \( C_{uv}^{w} \neq 0 \), it holds that

\[ p(v) - p(u) \geq Q_{uv}. \]

By using Lemma 5.3, we obtain the following lemma.

**Lemma 5.4.** Suppose that \( p \) and \( q \) are feasible dual variables. Let \( k \) be an integer and let \( X \subseteq V^+ \) be a vertex subset such that \( X \cap H_i = \emptyset \) for any \( i > k \) and \( C^k[X] \) is nonsingular. Then, we have

\[ p(X \setminus B^*) - p(X \cap B^*) \geq \sum_{i=1}^{\mu} \left( \sum_{j \in J_{uv}} q(H_i) - \sum_{i \in I_{uv} \cap J_{uv}} q(H_i) \right). \]

Proof. Since \( C^k[X] \) is nonsingular, there exists a perfect matching \( M = \{(u_j, v_j) \mid j = 1, \ldots, \mu\} \) between \( X \setminus B^* \) and \( X \cap B^* \) such that \( u_j \in X \cap B^*, v_j \in X \setminus B^* \) and \( C_{u_jv_j}^k \neq 0 \) for \( j = 1, \ldots, \mu \). Since \( X \cap H_i = \emptyset \) for any \( i > k \) implies that \( i_{uv} \leq k \), by Lemma 5.3, we have

\[ p(v_j) - p(u_j) \geq Q_{u_jv_j} \]

for \( j = 1, \ldots, \mu \). By combining these inequalities, we obtain

\[ p(X \setminus B^*) - p(X \cap B^*) \geq \sum_{j=1}^{\mu} \left( \sum_{i \in I_{u_jv_j} \setminus J_{u_jv_j}} q(H_i) - \sum_{i \in I_{u_jv_j} \cap J_{u_jv_j}} q(H_i) \right). \]

It suffices to show that, for each \( i \), the coefficient of \( q(H_i) \) in the right hand side of (1) is

- at least \( -1 \) if \( H_i \in \Lambda_n \) and \( H_i \cap H_j = \emptyset \)
- at least \( 1 \) if \( H_i \in \Lambda_n \) and \( H_i \cap H_j \) is odd, or \( t_i \notin X \), and \( t_i \in H_j \in \Lambda_n \) and \( X \cap H_i \) is odd, or \( t_i \notin X \), and \( t_i \in H_j \in \Lambda_n \) and \( X \cap H_i \) is odd.
- at least \( 0 \) if \( X \cap H_i \) is even.

For each \( i \), since \( i \in I_{u_jv_j} \cap J_{u_jv_j} \) implies \( t_j \in \{u_j, v_j\} \), there exists at most one index \( j \) such that \( i \in I_{u_jv_j} \cap J_{u_jv_j} \). This shows that the coefficient of \( q(H_i) \) in (1) is at least \(-1\).

Suppose that either (i) \( H_i \in \Lambda_n \) and \( X \cap H_i \) is odd, or (ii) \( H_i \in \Lambda_n \) and \( X \cap H_i \) is odd. In both cases, there is no index \( j \) with \( i \in I_{u_jv_j} \cap J_{u_jv_j} \). Furthermore, since \( X \cap H_i \) is odd, there exists an index \( j' \) such that \( i \in I_{u_jv_j} \cap J_{u_jv_j} \), which shows that the coefficient of \( q(H_i) \) in (1) is at least 1.

If \( X \cap H_i \) is even, then there exist an even number of indices \( j \) such that \( (u_j, v_j) \) crosses \( H_i \). Therefore, if there exists an index \( j \) such that \( i \in I_{u_jv_j} \cap J_{u_jv_j} \), then there exists another index \( j' \) such that \( i \in I_{u_jv_j} \cap J_{u_jv_j} \). Thus, the coefficient of \( q(H_i) \) in (1) is at least 0 if \( X \cap H_i \) is even.

We consider the tightness of the inequality in Lemma 5.4. For \( k = 0, 1, \ldots, \lambda \), let \( C^k = (V^*, F^k) \) be the graph such that \( (u, v) \in F^k \) if and only if \( C_{uv}^k \neq 0 \) (or \( C_{uv}^k = 0 \)). An edge \((u, v) \in F^k \) with \( u \in B^+ \) and \( v \in V^+ \setminus B^+ \) is said to be tight if \( p(v) - p(u) = Q_{uv} \). We
say that a matching $M \subseteq F^k$ is consistent with a blossom $H_i \in \Lambda$ if one of the following three conditions holds:

- $H_i \in \Lambda_n$, and $\{(u, v) \in M \mid i \in I_{uv}\} \subseteq 1$.
- $H_i \in \Lambda_n$, $ti \notin \partial M$, and $\{(u, v) \in M \mid i \in I_{uv}\} \subseteq 1$.
- $H_i \in \Lambda_n$, $ti \in \partial M$, and $\{(u, v) \in M \mid i \in I_{uv} \cap J_{uv}\} \subseteq 1$.

Here, $\partial M$ denotes the set of the end vertices of $M$. For $k \in \{1, \ldots, \lambda\}$, we say that a matching $M \subseteq F^k$ is tight if every edge of $M$ is tight and $M$ is consistent with every positive blossom $H_i$. As the proof of Lemma 5.4 clarifies, if there exists a tight perfect matching $M$ in the subgraph $G^k[X]$ of $G^k$ induced by $X$, then the inequality of Lemma 5.4 is tight. Furthermore, in such a case, every perfect matching in $G^k[X]$ must be tight, which is stated as follows.

**Lemma 5.5.** For $k \in \{0, 1, \ldots, \lambda\}$ and a vertex set $X \subseteq V^*$, if $C^k[X]$ has a tight perfect matching, then any perfect matching in $C^k[X]$ is tight.

### 6 Optimality

In this section, we show that if we obtain a parity base $B$ and feasible dual variables $p$ and $q$, then $B$ is a minimum-weight parity base.

Note again that although $p$ and $q$ are called dual variables, they do not correspond to dual variables of an LP-relaxation of the minimum-weight parity base problem. Our optimality proof is based on the algebraic formulation of the problem (Lemma 2.1) and the duality of the maximum weight matching problem.

**Theorem 6.1.** If $B : = B^* \cap V$ is a parity base and there exist feasible dual variables $p$ and $q$, then $B$ is a minimum-weight parity base.

**Proof.** Since the optimal value of the minimum-weight parity base problem is represented with $\deg Pf \Phi_A(\theta)$ as shown in Lemma 2.1, we evaluate the value of $\deg Pf \Phi_A(\theta)$, assuming that we have a parity base $B$ and feasible dual variables $p$ and $q$.

Recall that $A$ is transformed to $[I \mid C]$ by applying row transformations and column permutations, where $C$ is the fundamental circuit matrix with respect to the base $B$ obtained by $C = A[U, B]^{-1} A[U, V \setminus B]$. Note that the identity submatrix gives a one to one correspondence between $U$ and the row set of $C$ can be regarded as $U$. We now apply the same row transformations and column permutations to $\Phi_A(\theta)$, and then apply also the corresponding column transformations and row permutations to obtain a skew-symmetric polynomial matrix $\Phi_A(\theta)$, that is,

$$\Phi_A(\theta) = \begin{pmatrix} 0 & O & O \\ -I & C & D'(\theta) \\ -C^T & D'(\theta) & 0 \end{pmatrix} 
\begin{array}{c} U \\ B \\ V \setminus B \end{array}$$

where $D'(\theta)$ is in a block-diagonal form obtained from $D(\theta)$ by applying row and column permutations simultaneously. Note that $Pf \Phi_A(\theta) = \pm Pf \Phi_A(\theta) / \det A[U, B]$, where the sign is determined by the ordering of $V$.

We now define

$$\Phi_A(\theta) = \begin{pmatrix} O & O & C^*[V \cup T] \\ O & O & I \\ -I & D'(\theta) & O \end{pmatrix} 
\begin{array}{c} T \cap B^* \\ U \\ B \\ V \setminus B \end{array}$$

obtained from $\Phi_A(\theta)$ by attaching rows and columns corresponding to $T := \{t_i, \bar{t}_i \mid i \in \{1, \ldots, \lambda\}\}$. Note that $t_i$ and $\bar{t}_i$ do exist for each $i$, as there is no source line and hence $\Lambda = \Lambda_n$. The row and column sets of $\Phi_A(\theta)$ are both indexed by $W^* := V \cup U \cup T$. By the definition of $\Phi_A(\theta)$, we have $\Phi_A(\theta)i_{uv} = 0$ for $v \in W^* \setminus \{t_i\}$ and $(\Phi_A(\theta))i_{t_i\bar{t}_i}$ is a nonzero constant, which shows that $\deg Pf \Phi_A(\theta) = \deg Pf \Phi_A(\theta)$.

Recall that $C^1$ is obtained from $C^*$ by adding a row (resp. column) corresponding to $t_i$ to another row (resp. column) repeatedly. By applying the same transformation to $\Phi_A(\theta)$, we obtain the following matrix:

$$\Phi_A(\theta) = \begin{pmatrix} O & O & C^*[V \cup T] \\ O & O & I \\ -I & D'(\theta) & O \end{pmatrix}$$

Note that $Pf \Phi_A(\theta) = Pf \Phi_A(\theta)$. Thus we have $\deg Pf \Phi_A(\theta) = \deg Pf \Phi_A(\theta)$.

Construct a graph $G^* = (W^*, E^*)$ with edge set $E^*$ defined by $E^* = \{(u, v) \mid (\Phi_A(\theta))_{uv} \neq 0\}$. Each edge $(u, v) \in E^*$ has a weight $\deg Pf \Phi_A(\theta)_{uv}$. Then it can be easily seen that the maximum weight of a perfect matching in $G^*$ is at least $\deg Pf \Phi_A(\theta) = \deg Pf \Phi_A(\theta)$. Let us recall that the dual linear program of the maximum weight perfect matching problem on $G^*$ is formulated as follows.

Minimize $\sum_{v \in W^*} \pi(v) - \sum_{Z \in \Omega} \xi(Z)$
subject to $\pi(u) + \pi(v) - \sum_{Z \in \Omega_{uv}} \xi(Z) \geq 0$ for $(Z \in \Omega)$,

where $\Omega = \{Z \mid Z \subseteq W^*, |Z| \text{ odd}, |Z| \geq 3\}$ and $\Omega_{uv} = \{Z \mid Z \in \Omega, \{u, v\} \subseteq Z\}$ (see e.g. [31, Theorem 25.1]). In what follows, we construct a feasible solution $(\pi, \xi)$ of this linear program. The objective value provides an upper bound on the maximum weight of a perfect matching in $G^*$, and consequently serves as an upper bound on $\deg Pf \Phi_A(\theta)$.

Since $\Phi_A(\theta)(U, B)$ is the identity matrix, we can naturally define a bijection $\beta : B \rightarrow U$ between $B$ and $U$. For $v \in U \cup (T \cap B^*)$, let $v'$ be the vertex in $V^*$ that corresponds to $v$, that is, $\beta_v = \beta^{-1}(v)$ if $v \in U$ and $\beta_v = v$ if $v \in T \cap B^*$. We define $\pi' : W^* \rightarrow \mathbb{R}$ by

$$\pi'(v) = \begin{cases} \pi(v) & \text{if } v \in V \cup (T \setminus B^*), \\
-\pi(v) & \text{if } v \in U \cup (T \cap B^*), 
\end{cases}$$

and define $\pi : W^* \rightarrow \mathbb{R}$ by

$$\pi(v) = \begin{cases} \pi'(v) + q(h_i) & \text{if } v = t_i \text{ or } v' = t_i \text{ for some } i, \\
\pi'(v) & \text{otherwise.} 
\end{cases}$$

For $i \in \{1, \ldots, \lambda\}$, let $Z_i = ((H_i \cap (V \cup T)) \cup \beta(H_i \cap B) \setminus \{t_i\}$ and define $\xi(Z_i) = q(h_i)$. See Fig. 3 for an example. For any $i \in \{1, \ldots, \lambda\}$, since $H_i \cap (V \cup T)$ consists of lines and dummy lines, and there is no source line in $G$, we see that both $|H_i \cap (V \cup T)|$ and $|\beta(H_i \cap B)|$ are even, which shows that $|Z_i|$ is odd and $|Z_i| \geq 3$. Satoru Iwata and Yusuke Kobayashi
Define $\xi(Z) = 0$ for any $Z \in \Omega \setminus \{Z_1, \ldots, Z_\lambda\}$. We now show the following claim.

![Diagram](image)

Figure 3: Definition of $Z_i$. Lines and dummy lines are represented by double bonds.

**CLAIM 6.2. The dual variables $\pi$ and $\xi$ defined as above form a feasible solution of the linear program.**

**Proof.** Suppose that $(u, v) \in E^*$. If $u, v \in V$ and $u = \bar{v}$, then (DF1) shows that $\pi(u) + \pi(v) = p(\bar{v}) + p(v) = w_\ell = \deg_B(\Phi^1_A(\emptyset))_{uv}$, where $\ell = \{u, \bar{v}\}$. Since $|Z_i \cap \{u, \bar{v}\}|$ is even for any $i \in \{1, \ldots, \lambda\}$, this shows (2). If $u \in U$ and $v \in B$, then $(u, v) \in E^*$ implies that $u = \beta(\bar{v})$, and hence $\pi(u) + \pi(v) = 0$, which shows (2) as $|Z_i \cap \{u, \bar{v}\}|$ is even for any $i \in \{1, \ldots, \lambda\}$.

The remaining case of $(u, v) \in E^*$ is when $u \in U \cup (T \cap B^*)$ and $v \in (V \setminus B) \cup (T \setminus B^*)$. That is, it suffices to show that $(u, v)$ satisfies (2) if $\xi_{uv} = 0$. Recall that $u^*$ is the vertex in $V^*$ that corresponds to $u$. By the definition of $\pi$, we have

$$\pi(u) + \pi(v) = p(v) - p(u^*) + \sum_{i \in J_{uv}} q(H_i). \tag{3}$$

By the definition of $Z_i$, we have $|Z_i \cap \{u, v\}| = 1$ if and only if $i \in I_{u^*} \cup J_{uv}$, which shows that

$$\sum_{i \in I_{u^*} \cup J_{uv}} \xi(Z_i) = \sum_{i \in I_{u^*} \cup J_{uv}} q(H_i) + \sum_{i \in J_{uv} \cup J_{uv}^\circ} q(H_i). \tag{4}$$

Since $\xi^1_{uv} \neq 0$, by Lemma 5.3, we have

$$p(v) - p(u^*) \geq Q_{uv} = \sum_{i \in I_{uv} \cup J_{uv}} q(H_i) - \sum_{i \in J_{uv} \cup J_{uv}^\circ} q(H_i). \tag{5}$$

By combining (3), (4), and (5), we obtain

$$\pi(u) + \pi(v) = \sum_{i \in I_{u^*} \cup J_{uv}} \xi(Z_i) \geq 0,$$

which shows that $(u, v)$ satisfies (2). \hfill \Box

The objective value of this feasible solution is

$$\sum_{v \in W^*} \pi(v) - \sum_{i \in \{1, \ldots, \lambda\}} \xi(Z_i) = \sum_{v \in W^*} \pi(v) = \sum_{\ell \subseteq V \setminus B} w_\ell \tag{6}$$

where the first equality follows from the definition of $\pi$ and $\xi$, the second one follows from the definition of $\pi^*$ and the fact that $p(\ell) = p(\ell_i)$ for each $i$, and the third one follows from (DF1). By the weak duality of the maximum weight matching problem, we have

$$\sum_{v \in W^*} \pi(v) - \sum_{i \in \{1, \ldots, \lambda\}} \xi(Z_i) \geq (\text{maximum weight of a perfect matching in } G^\ast) \geq \deg_B \Phi^A(\emptyset) = \deg_B \Phi^A(\emptyset). \tag{7}$$

On the other hand, Lemma 2.1 shows that any parity base $B'$ satisfies that

$$\sum_{\ell \subseteq B'} w_\ell \geq \deg_B \Phi^A(\emptyset), \tag{8}$$

Combining (6)–(8), we have $\sum_{\ell \subseteq V \setminus B} w_\ell = \deg_B \Phi^A(\emptyset)$, which means $B$ is a minimum-weight parity base by Lemma 2.1. \hfill \Box

## 7 FINDING AN AUGMENTING PATH

In this section, we define an augmenting path and present a procedure for finding one. The validity of our procedure is shown in Section 8 and the full version [14].

Suppose we are given $V^*, B^*, C^*, \lambda$, and feasible dual variables $p$ and $q$. Recall that, for $i = 0, 1, \ldots, \lambda$, we denote by $G^i = (V^*, F^i)$ the graph with edge set $F^i := \{(u, v) \mid C_{uv} \neq 0\}$. Since $C^0 = C^*$, we use $F^0$ instead of $F^0$. By Lemma 5.3, we have $p(v) - p(u) \geq Q_{uv}$ if $(u, v) \in F^\lambda$, $u \in B^*$, and $v \in V^* \setminus B^*$. Let $F^\lambda \subseteq F^\lambda$ be the set of tight edges in $F^\lambda$, that is, $F^\lambda = \{(u, v) \mid u \in B^* \setminus (\forall \ell \subseteq V \setminus B, p(v) - p(u) = Q_{uv})\}$. Our procedure works primarily on the graph $G^\lambda = (V^*, F^\lambda)$. For a vertex set $X \subseteq V^*$, $G^\lambda[X]$ (resp. $G^\lambda[X]$) denotes the subgraph of $G^\lambda$ (resp. $G^\lambda$) induced by $X$.

Each normal blossom $H_i \in \Lambda_n$ has a specified vertex $g_i \in H_i$, which we call a generator of $g_i$. When we search for an augmenting path, we keep the following properties of $g_i$ and $t_i$.

((GT1)) For each $H_i \in \Lambda_n$, there is no edge of $F^\lambda$ between $g_i$ and $V^* \setminus H_i$.

((GT2)) For each $H_i \in \Lambda_n$, there is no edge of $F^\lambda$ between $t_i$ and $H_i \setminus \{t_i, t_i\}$.

By (1-3) and (2-3) of Lemma 4.1, if $i \leq j \leq \lambda$, (GT1) implies that there is no edge of $F^\lambda$ between $g_i$ and $V^* \setminus H_i$. By (GT2), we can see that for each $H_i \in \Lambda_n$ and for $j = 0, 1, \ldots, \lambda$, there is no edge in $F^\lambda$ between $t_i$ and $H_i \setminus \{t_i, t_i\}$. Furthermore, since (GT2) implies that $C_{uv} = C_{uv}$ for each $i$ and $u, v \in H_i$, we have the following observation.

**Observation 7.1.** If (GT2) holds, then $F^\lambda$ coincides with $F^\lambda$ regardless of the ordering.

With this observation, it is natural to ask whether one can define the dual feasibility by using $F^\lambda$ instead of $F^\lambda$. However, (GT2) will be tentatively violated just after the augmentation, which is the reason why we use $F^\lambda$ in the definition of the dual feasibility.

Roughly, our procedure finds a part of the augmenting path outside the blossoms. The routing in each blossom $H_i$ is determined by a prescribed vertex set $R_{H_i}(x)$. For $i = 1, \ldots, \lambda$, define $H^n_i := (H_i \setminus \{t_i\}) \setminus \{t_i \mid H_i \in \Lambda_n\}$, where $\{t_i \mid H_i \in \Lambda_n\}$. For any $i \in \{1, \ldots, \lambda\}$ and for any $x \in H^n_i$, the prescribed vertex set $R_{H_i}(x) \subseteq H_i$ is assumed to satisfy the following.

((BR1)) $x \in R_{H_i}(x) \subseteq H_i \setminus \{t_i \mid H_i \in \Lambda_n\}$. 

270
We sometimes regard $R_{H_i}(x)$ as a sequence of vertices, and in such a case, the last two vertices are $xx$. We also suppose that the first two vertices are $t_it_i$ if $H_i \in \Lambda_n$ and the first vertex is the unique source vertex in $R_{H_i}(x)$ if $H_i \in \Lambda_n$. Each blossom $H_i \in \Lambda$ is assigned a total order $<_{H_i}$ among all the vertices in $H_i^*$.

We say that a vertex set $P \subseteq V^*$ is an augmenting path if it satisfies the following properties.

(P1) $P$ consists of normal lines, dummy lines, and two vertices from distinct source lines.

(P2) For each $H_i \in \Lambda$, either $P \cap H_i = \emptyset$ or $P \cap H_i = R_{H_i}(x_i)$ for some $x_i \in H_i^*$.

(P3) $G^o[P]$ has a unique tight perfect matching.

In the rest of this section, we describe how to find an augmenting path. Section 7.1 is devoted to the search procedure, which calls two procedures: RBlossom and DBlossom. Here, R and D stand for “regular” and “degenerate,” respectively. The details of these procedures are described in Section 7.2.

### 7.1 Search Procedure

In this subsection, we describe a procedure for searching for an augmenting path. The procedure performs the breadth-first search using a queue to grow paths from source vertices. A vertex $v \in V^*$ is labeled and put into the queue when it is reached by the search. The procedure picks the first labeled element from the queue, and examines its neighbors. A linear order $< \in \mathbb{R}$ is defined on the labeled vertex set so that $u < v$ means $u$ is labeled prior to $v$.

For each $x \in V^*$, we denote by $K(x)$ the maximal blossom that contains $x$. If a vertex $x \in V$ is not contained in any blossom, then it is called single and we denote $K(x) = \{x, x\}$. The procedure also labels some blossoms with $\oplus$ or $\ominus$, which will be used later for modifying dual variables. With each labeled vertex $v$, the procedure associates a path $P(v)$ and its subpath $J(v)$, where a path is a sequence of vertices. The first vertex of $P(v)$ is a labeled vertex in a source line and the last one is $v$. The reverse path of $P(v)$ is denoted by $P(\bar{v})$. For a path $P(v)$ and a vertex $r \in P(v)$, we denote by $P(v)[r]$ the subsequence of $P(v)$ after $r$ (not including $r$). We sometimes identify a path with its vertex set. When an unlabeled vertex $u$ is examined in the procedure, we assign a vertex $\rho(u)$ and a path $I(u)$. The procedure is described as follows.

**Procedure** Search

**Step 3:** While there exists a labeled vertex $u$ adjacent to $v$ in $G^o$ with $K(u) \neq K(v)$, choose such $u$ that is minimum with respect to $<$. Set $\rho(u)$ as follows and do the following steps (3-1) and (3-2).

(3-1) If the first elements in $P(v)$ and in $P(u)$ belong to different source lines, then return $P := P(v)P(u)$ as an augmenting path.

(3-2) Otherwise, apply RBlossom$(v, u)$ to add a new blossom to $\Lambda$.

**Step 4:** While there exists an unlabeled vertex $x$ adjacent to $v$ in $G^o$ such that $\rho(u)$ is not assigned, do the following steps (4-1)-(4-5).

(4-1) If $u$ is a single vertex and $(v, u) \notin E^o$, then label $u$ and $v$ with $P(u) := P(v)u$ and $J(u) := \emptyset$, set $\rho(u) := v$ and $I(u) := \{u\}$, and put $u$ into the queue.

(4-2) If $u$ is a single vertex and $(v, u) \notin E^o$, then apply DBlossom$(v, u)$.

(4-3) If $K(u) = H_i \in \Lambda_n$, $(v, t_i) \in E^o$, and $F^\lambda$ contains an edge between $v$ and $H_i \setminus \{t_i\}$, then apply DBlossom$(v, t_i)$.

(4-4) If $K(u) = H_i \in \Lambda_n$, $(v, t_i) \in E^o$, and $F^\lambda$ contains no edge between $v$ and $H_i \setminus \{t_i\}$, then label $H_i$ with $\ominus$, set $\rho(t_i) := v$ and $I(t_i) := \{t_i\}$, and do the following. For each unlabeled vertex $x \in H_i^\lambda$ in the order of $<_{H_i}$, label $x$ with $P(x) := P(v)R_{H_i}(x)$ and $J(x) := R_{H_i}(x) \setminus \{t_i\}$, and put $x$ into the queue.

(4-5) If $K(u) = H_i \in \Lambda_n$ and $(v, t_i) \notin E^o$, then choose $y \in H_i \setminus \{t_i\}$ with $(v, y) \notin E^o$ that is minimum with respect to $<_{H_i}$, and do the following. Label $H_i$ with $\ominus$, label $t_i$ with $P(t_i) := P(v)R_{H_i}(y)$ and $J(t_i) := \{t_i\}$, and put $t_i$ into the queue. For each unlabeled vertex $x \in H_i^\lambda$, set $\rho(x) := v$ and $I(x) := R_{H_i}(x) \setminus \{t_i\}$.

**Step 5:** Go back to Step 2.

### 7.2 Creating a Blossom

In this subsection, we describe two procedures that create a new blossom. The first one is RBlossom called in Step (3-2) of Search.

**Procedure** RBlossom$(v, u)$

**Step 1:** Let $c$ be the last vertex in $P(u)$ such that $K(c)$ contains a vertex in $P(u)$. Let $d$ be the last vertex in $P(u)$ contained in $K(c)$. Note that $K(c) = K(d)$. If $c = d$, then define $Y := \bigcup \{K(x) \mid x \in P(v) \cup P(u)\} \cup \{c\}$ and $r := c$. Otherwise, define $Y := \bigcup \{K(x) \mid x \in P(v) \cup P(u)\}$ and $r := c$. Otherwise, let $r$ be the last vertex in $P(v)$ not contained in $Y$ if exists. See Fig. 4 for an example.

**Step 2:** If $Y$ contains no source line, then define $g$ to be the vertex subsequent to $r$ in $P(v)$ and introduce new vertices $b, t, \bar{t}, \bar{b}$ (see below for the details).

**Step 3:** Define $H := Y \cup \{b, t, \bar{t}\}$ if $Y$ contains no source line, and $H := Y$ otherwise.

**Step 4:** If $H$ contains no source line, then for each labeled vertex $x$ with $P(x) \cap H \neq \emptyset$, replace $P(x)$ by $P(x) \setminus P(r)P(x)[r]$. Label $I$ with $P(I) := P(r)I$ and $J(I) := \{t\}$, and extend the ordering $< \in \mathbb{R}$ of the labeled vertices so that $I$ is just after $r$, i.e., $r < I$ and no element is between $r$ and $I$. For each vertex $x \in H$ with $\rho(x) = r$, update $\rho(x)$ as $\rho(x) := I$. Set $\rho(t) := r$ and $I(t) := \{t\}$.
If we consider the following two cases separately.

For two unlabeled vertices $x, y \in H^\circ$, the procedure explicitly, but we introduce them to show the validity for some.

\begin{align*}
\hat{V}^* = V^* \cup \{b, t, l\}, \quad \hat{B}^* = B^* \cup \{l, t\}, \quad \text{and let } \hat{p} : \hat{V}^* \to \mathbb{R} \text{ be an extension of } p \text{ such that } \hat{p}(b) = \hat{p}(t) = \hat{p}(l) = p(r) + Q_{rb}.
\end{align*}

If $r \in V^* \setminus B^*$ and $g \in B^*$, then define $\hat{V}^*, \hat{B}^*, \hat{C}^\lambda$, and $\hat{p}$ as follows.

\begin{itemize}
  \item \(\hat{V}^* = V^* \cup \{b, t, l\}\), \(\hat{B}^* = B^* \cup \{l, t\}\), and let \(\hat{p} : \hat{V}^* \to \mathbb{R}\) be an extension of \(p\) such that \(\hat{p}(b) = \hat{p}(t) = \hat{p}(l) = p(r) + Q_{rb}\).
  \item \(\hat{C}^\lambda_{by} = C^\lambda_{by}\) for any \(y \in V^* \setminus B^*\) and \(\hat{C}^\lambda_{by} = 0\) for any \(y \in (V^* \setminus B^*) \setminus \{t\}\).
  \item \(\hat{C}^\lambda_{xt} = C^\lambda_{xt}\) for any \(x \in (B^* \setminus Y) \cup \{b\}\) and \(\hat{C}^\lambda_{xt} = 0\) for any \(x \in B^* \cap Y\).
\end{itemize}

Then, we rename $\hat{V}^*, \hat{B}^*, \hat{C}^\lambda$, and $\hat{p}$ to $V^*, B^*, C^\lambda$, and $p$, respectively.

The next one is DBlossom, called in Steps (4-2) and (4-3) of Search.

**Procedure DBlossom\((v, u)\)**

**Step 1:** Set $Y := K(u), r := v$, and $g := u$. Introduce new vertices $b, t, l$ in the same way as Step 2 of DBlossom\((u, v)\), and define $H := Y \cup \{b, t, l\}$. Label $t$ with $P(t) := P(v)t\hat{t}$ and $J(t) := \{l\}$, and extend the ordering $<_{H}$ of the labeled vertices so that $t$ is just after $v$, i.e., $v < t$ and no element is between $v$ and $t$. Set $\rho(t) := v$ and $I(t) := \{l\}$.

**Step 2:** If $Y = H_0$ for some positive blossom $H_0 \in \Lambda$, then do the following. For each vertex $x \in H_0$, label $x$ with $P(x) := P(v)t\hat{t}x\hat{x}$ and $J(x) := \hat{x}x\hat{t}$, and put $x$ into the queue.

**Step 3:** Let $H$ with $\hat{t}$. Define $R_H(x) := P(x)v$ for each $x \in H^\circ$. Define $<_{H}$ by the ordering $<_{H}$ of the labeled vertices in $H^\circ$. Add $H$ to $\Lambda$ with $q(H) = 0$ regarding $b, t, l, \lambda$, and $\gamma$, as the bud of $H$, and $\rho$, and the order of $H$, respectively, and update $H, A, C, G, \lambda, \gamma$, and $K(v)$ for $v \in V^*\setminus H$, accordingly.

**Step 4:** If $Y = H_0$ for some positive blossom $H_0 \in \Lambda$, then set $\epsilon := q(H_0)$ and modify the dual variables as follows: $q(H) := q(H) - \epsilon, q(H) := q(H) + \epsilon$.

\begin{align*}
\rho(t) := \begin{cases} p(t) - \epsilon & \text{if } t \in V^* \setminus B^*, \\
p(t) + \epsilon & \text{if } t \in B^* \\
p(t) & \text{if } t \in V^* \setminus B^*
\end{cases}
\end{align*}

Since $q(H_0)$ becomes zero, we delete $H_0$ from $\Lambda$ as shown in Lemma 5.2. We also remove $b, t, l$, and from $V^*$ and update $H, A, C, G, \lambda, \gamma$, and $K(v)$ for $v \in V^n$, accordingly.

We note that Step 4 of DBlossom\((u, v)\) is executed to keep the condition $H_0 \cap V \neq H_0 \cap V$ for distinct $H_0, H_j \in \Lambda$.

### 8 VALIDITY

The procedures described in Section 7 are designed so that they keep the conditions (GT1), (GT2), (BT), (DF1)–(DF3), and (BR1)–(BR3). Assuming these conditions, we can show that a nonempty output of Search is indeed an augmenting path. See the full version [14] for details.
9 DUAL UPDATE

In this section, we describe how to modify the dual variables when Search returns \( \emptyset \) in Step 2. In Section 9.1, we show that the procedure keeps the dual variables finite as long as the instance has a parity base. In Section 9.2, we bound the number of dual updates per augmentation.

Let \( R \subseteq V^+ \) be the set of vertices that are reached or examined by the search procedure and not contained in any blossoms, i.e., \( R = R^+ \cup R^- \), where \( R^+ \) is the set of labeled vertices that are not contained in any blossom, and \( R^- \) is the set of unlabeled vertices whose mates are in \( R^+ \). Let \( Z \) denote the set of vertices in \( V^+ \) contained in labeled blossoms. The set \( Z \) is partitioned into \( Z^+ \) and \( Z^- \), where

\[
Z^+ = \{ t_i \mid H_i: a \text{ maximal blossom labeled with } \emptyset \} \cup \bigcup \{ (H_i \setminus \{ t_i \}) \mid H_i: a \text{ maximal blossom labeled with } \emptyset \},
\]

\[
Z^- = \{ t_i \mid H_i: a \text{ maximal blossom labeled with } \emptyset \} \cup \bigcup \{ (H_i \setminus \{ t_i \}) \mid H_i: a \text{ maximal blossom labeled with } \emptyset \}.
\]

We denote by \( Y \) the set of vertices that do not belong to these subsets, i.e., \( Y = V^+ \setminus (R \cup Z) \).

For each vertex \( v \in R \), we update \( p(v) \) as

\[
p(v) := \begin{cases} p(v) + \epsilon & (v \in R^+ \cap B^-) \\ p(v) - \epsilon & (v \in R^- \cap B^-) \\ p(v) - \epsilon & (v \in R^+ \cap B^+) \\ p(v) + \epsilon & (v \in R^- \cap B^+). \end{cases}
\]

We also modify \( q(H) \) for each maximal blossom \( H \) by

\[
q(H) := \begin{cases} q(H) + \epsilon & (H: \text{ labeled with } \emptyset) \\ q(H) - \epsilon & (H: \text{ labeled with } \emptyset) \\ q(H) & (\text{otherwise}). \end{cases}
\]

To keep the feasibility of the dual variables, \( \epsilon \) is determined by

\[
\epsilon = \min \{ \epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4 \},
\]

where

\[
\epsilon_1 = \frac{1}{2} \min \{ p(u) - p(v) - Q_{uv} \mid (u, v) \in F_{\ell}, u, v \in R^+ \cup Z^+, K(u) \neq K(v) \},
\]

\[
\epsilon_2 = \min \{ p(u) - p(v) - Q_{uv} \mid (u, v) \in F_{\ell}, u \in R^+ \cup Z^+, v \in Y \},
\]

\[
\epsilon_3 = \min \{ p(u) - p(v) - Q_{uv} \mid (u, v) \in F_{\ell}, u \in R^- \cup Z^+, v \in Y \},
\]

\[
\epsilon_4 = \min \{ q(H) \mid H: a \text{ maximal blossom labeled with } \emptyset \}.
\]

We note that \( F_{\ell} \) coincides with \( F^\lambda \) as seen in Observation 7.1. If \( \epsilon = +\infty \), then we terminate Search and conclude that there exists no parity base. If there are any blossoms whose values of \( q \) become zero, then the algorithm deletes those blossoms from \( \Lambda \), which is possible by Lemma 5.2. Then, apply the procedure Search again.

9.1 Detecting Infeasibility

By the definition of \( \epsilon \), we can easily see that the updated dual variables are feasible if \( \epsilon \) is a finite value. We now show that we can conclude that the instance has no parity base if \( \epsilon = +\infty \).

A skew-symmetric matrix is called an alternating matrix if all the diagonal entries are zero. Note that any skew-symmetric matrix is alternating unless the underlying field is of characteristic two. By a congruence transformation, an alternating matrix can be brought into a block-diagonal form in which each nonzero block is a \( 2 \times 2 \) alternating matrix. This shows that the rank of an alternating matrix is even, which plays an important role in the proof of the following lemma.

**Lemma 9.1.** Suppose that there is a source line, and suppose also that \( \epsilon = +\infty \) when we update the dual variables. Then, the instance has no parity base.

**Proof.** Recall that \( Y \) is the set of vertices \( v \) such that \( K(v) \) contains no labeled vertices. In the proof, we use the following properties of \( F^\lambda \):

(A) there exists no edge in \( F^\lambda \) between two labeled vertices \( u, v \in V^+ \) with \( K(u) \neq K(v) \), and

(B) there exists no edge in \( F^\lambda \) between a labeled vertex in \( V^+ \setminus Y \) and a vertex in \( Y \),

but do not use the dual feasibility. Therefore, we may assume that \( Y \) contains no blossom, because removing such blossoms from \( \Lambda \) does not create a new edge in \( F^\lambda \) between a labeled vertex in \( V^+ \setminus Y \) and a vertex in \( Y \). Note that this operation might violate the dual feasibility. Let \( \Lambda_{\text{max}} \subseteq \Lambda \) be the set of maximal blossoms. Since \( \epsilon_4 = +\infty \), any blossom \( H_i \in \Lambda_{\text{max}} \) is labeled with \( \emptyset \). Let \( L_n \) be the set of source lines that are not contained in any blossom. Let \( L_n \) be the set of normal lines \( \ell \) such that \( \ell \) is not contained in any blossom and \( \ell \) contains a labeled vertex. We can see that for each line \( \ell \in L_n \), exactly one vertex \( v_\ell \) in \( \ell \) is unlabeled and the other vertex \( v_\ell^\prime \) is labeled.

In order to show that there is no parity base, by Lemma 2.1, it suffices to show that \( \Phi_{\Lambda}(\emptyset) = 0 \). We construct the matrix

\[
\Phi_{\Lambda}^\lambda(\emptyset) = \begin{pmatrix} O & O & C^\lambda & V \cup T \end{pmatrix} \begin{pmatrix} T \cap B^* \end{pmatrix} \begin{pmatrix} \Phi_{\Lambda}(\emptyset) \end{pmatrix} \begin{pmatrix} U \end{pmatrix} \begin{pmatrix} B \end{pmatrix} \begin{pmatrix} V \setminus B \end{pmatrix} \begin{pmatrix} T \cap B^* \end{pmatrix} \begin{pmatrix} O \end{pmatrix} \begin{pmatrix} O \end{pmatrix}
\]

in the same way as Section 6, where \( T := \{ t_i, i_1 \mid H_i \in \Lambda_n \} \). Since \( \Phi_{\Lambda}(\emptyset) = 0 \) is equivalent to \( \Phi_{\Lambda}^\lambda(\emptyset) = 0 \), it suffices to show that \( \Phi_{\Lambda}^\lambda(\emptyset) \) is singular, i.e., rank \( \Phi_{\Lambda}^\lambda(\emptyset) \) is \( |U| + |V| + |T| \).

In order to evaluate rank \( \Phi_{\Lambda}^\lambda(\emptyset) \), we consider a skew-symmetric matrix \( \Phi_{\Lambda}(\emptyset) \) obtained from \( \Phi_{\Lambda}^\lambda(\emptyset) \) by attaching rows and column by applying row and column transformations as follows.

- For each line \( \ell \in L_n \), regard \( v_\ell \) as a vertex in \( U \) if \( \ell \subseteq B \) and regard \( v_\ell \) as a vertex in \( V \setminus B \) if \( \ell \subseteq V \setminus B \). Add a row and a column indexed by a new element \( z_\ell \) such that \( \Phi_{\Lambda}(\emptyset)_{z_\ell u} = -\Phi_{\Lambda}(\emptyset)_{z_\ell v} = 0 \) if \( u \neq v \) and \( \Phi_{\Lambda}(\emptyset)_{z_\ell z_\ell} = 1 \). Then, sweep out nonzero entries \( \Phi_{\Lambda}(\emptyset)_{z_\ell u} \) and \( \Phi_{\Lambda}(\emptyset)_{xz_\ell} \) with \( x \neq z_\ell \) using the row and the column indexed by \( z_\ell \).

- For each blossom \( H_i \in \Lambda_{\text{max}} \cap \Lambda_n \), add a row and a column indexed by a new element \( z_i \) such that \( \Phi_{\Lambda}(\emptyset)_{zz_i} = -\Phi_{\Lambda}(\emptyset)_{z_\ell v} = 0 \) if \( u \neq z_\ell \) and \( \Phi_{\Lambda}(\emptyset)_{zz_i} = -\Phi_{\Lambda}(\emptyset)_{z_\ell z_\ell} = 1 \). Then, sweep out nonzero entries \( \Phi_{\Lambda}(\emptyset)_{z_\ell z_i} \) and \( \Phi_{\Lambda}(\emptyset)_{xz_i} \) with \( x \neq z_i \) using the row and the column indexed by \( z_i \).

Note that we apply the above operations for each \( \ell \in L_n \) and for each \( H_i \in \Lambda_{\text{max}} \cap \Lambda_n \) in an arbitrary order.
rows and columns does not decrease the rank of a matrix, we have \( \text{rank } \Phi_A(\theta) \geq \text{rank } \Phi^\theta_A(\theta) \). The above operations sweep out all nonzero entries \( C_{xy}^\theta \) if either \( x \) or \( y \) is unlabeled. This together with (A) and (B) shows that \( \Phi_A(\theta) \) is a block-diagonal skew-symmetric matrix, where the index set of each block corresponds to one of the following vertex sets: (i) \( \ell \cup \{ z \} \) for \( \ell \in L_n \), (ii) \( \ell \in L_n \), (iii) \( (H_t \cap (V \cup T)) \cup \{ z \} \) for \( H_t \in \Lambda_{\max} \cap \Lambda_n \), (iv) \( H_t \cap (V \cup T) \) for \( H_t \in \Lambda_{\max} \cap \Lambda_s \), or (v) \( Y \). Note that a vertex in \( B \) corresponds to two indices (i.e., two rows and two columns) of \( \Phi_A(\theta) \), where one is in \( B \) and the other is in \( U \), and a vertex in \( (V \setminus B) \cup T \) corresponds to one index of \( \Phi_A(\theta) \). We denote this partition of the index set by \( V_1, \ldots, V_k \). Then, we have

\[
\text{rank } \Phi_A(\theta) = \sum_{j = 1}^k \text{rank } \Phi_A(\theta)[V_j],
\]

where \( \Phi_A(\theta)[V_j] \) is the principal submatrix of \( \Phi_A(\theta) \) whose rows and columns are both indexed by \( V_j \). In what follows, we evaluate \( \text{rank } \Phi_A(\theta)[V_j] \) for each \( j \).

If \( V_j \) corresponds to (i) \( \ell \cup \{ z \} \) for \( \ell \in L_n \), (ii) \( \ell \in L_n \), (iii) \( (H_t \cap (V \cup T)) \cup \{ z \} \) for \( H_t \in \Lambda_{\max} \cap \Lambda_n \), or (iv) \( H_t \cap (V \cup T) \) for \( H_t \in \Lambda_{\max} \cap \Lambda_s \), then we have that \( |V_j| \) is odd. Since \( \Phi_A(\theta)[V_j] \) is an alternating matrix, this implies that \( \text{rank } \Phi_A(\theta)[V_j] \leq |V_j| \). If \( V_j \) corresponds to \( Y \), then \( \text{rank } \Phi_A(\theta)[V_j] \leq |V_j| \). Hence, we have that

\[
\text{rank } \Phi_A(\theta) \\
\leq \sum_{j = 1}^k \text{rank } \Phi_A(\theta)[V_j] \\
\leq \sum_{j = 1}^k |V_j| - (k - 1) \\
\leq 2|B| + |V \setminus B| + |T| + |L_n| + |\Lambda_{\max} \cap \Lambda_n| - (k - 1) \\
= |U| + |V| + |T| - |L_n| - |\Lambda_{\max} \cap \Lambda_s|.
\]

We note that \( |L_n| + |\Lambda_{\max} \cap \Lambda_s| \) is equal to the number of source lines. Therefore, since there exists at least one source line, we have that \( \text{rank } \Phi_A(\theta) < |U| + |V| + |T| \). Thus, \( \text{PF } \Phi_A(\theta) = \text{PF } \Phi^\theta_A(\theta) = 0 \) and there is no parity base by Lemma 2.1.

**9.2. Bounding Iterations**

We next show that the dual variables are updated \( O(n) \) times per augmentation. To see this, roughly, we show that this operation increases the number of labeled vertices. Although Search contains flexibility on the ordering of vertices, it does not affect the set of the labeled vertices. This is guaranteed by the following lemma.

**Lemma 9.2.** A vertex \( x \in V^* \setminus \{ b_i \mid H_i \in \Lambda_u \} \) is labeled in Search if and only if there exists a vertex set \( X \subseteq V^* \) such that (i) \( X \cup \{ x \} \) consists of normal lines, dummy lines, and a source vertex \( x \), (ii) \( \text{PF } C^X[X] \) is nonsingular, and (iii) the following equality holds:

\[
p(X \setminus B^*) - p(X \cap B^*) = -\sum_{H_t \in \Lambda_n} \text{q}(H_t) |H_t \in \Lambda_n, |X \cap H_t| \text{ is odd}, t_i \in X| + \\
\sum_{H_t \in \Lambda_n, |X \cap H_t| \text{ is odd}, t_i \notin X} \text{q}(H_t) + \\
\sum_{H_t \in \Lambda_n, |X \cap H_t| \text{ is odd}} \text{q}(H_t) \ (H_t \in \Lambda_n, |X \cap H_t| \text{ is odd}).
\] (9)

**Proof.** If \( x \) is labeled in Search, then we obtain \( p(x) \) and \( X := P(x) \setminus \{ x \} \) satisfies the conditions by the arguments in Section 8 (see the full version [14]).

Suppose that \( X \) satisfies the above conditions, and assume to the contrary that \( x \) is not labeled. If \( x \) is not labeled, then we can update the dual variables keeping the dual feasibility. We now see how the dual update affects (9).

- If \( x \) is not contained in any blossom, then the left hand side of (9) decreases by \( e \) by updating \( p(s) \). Otherwise, the right hand side of (9) increases by \( e \) by updating \( q(K(s)) \).
- If \( x \) is not contained in any blossom, then the left hand side of (9) decreases by \( e \) or does not change by updating \( p(x) \). Otherwise, the right hand side of (9) increases by \( e \) or does not change by updating \( q(K(x)) \).
- Updating the other dual variables does not affect the equality (9), since \( s, x \notin H_t \) implies that \( |X \cap H_t| \) is even.

By combining these facts, after updating the dual variables, we have that the left hand side of (9) is strictly less than its right hand side, which contradicts Lemma 5.4.

By using this lemma, we bound the number of dual updates as follows.

**Lemma 9.3.** If there exists a parity base, then the dual variables are updated at most \( O(n) \) times before Search finds an augmenting path.

**Proof.** Suppose that we update the dual variables more than once, and we consider how the value of

\[
k(V^*, \Lambda) := |\{ v \in V^* \mid v \text{ is labeled} \} + |\Lambda_n| - 2|H_t \in \Lambda_n \mid H_t^\theta \text{ contains no labeled vertex}|.
\]

will change between two consecutive dual updates. Lemma 9.2 shows that, if \( v \in V^* \) is labeled at the time of the first dual update, then it is labeled again at the time of the second dual update unless \( v \) is removed in the procedure. Note that if a labeled vertex \( v \in V^* \) is removed, then \( v = i_t, H_t \in \Lambda_n \) is a maximal blossom labeled with \( \emptyset \), and \( q(H_t) = e \), which shows that \( |[H_t \in \Lambda_n \mid H_t^\theta \text{ contains no labeled vertex}]| \) will decrease by one. We also observe that if a new blossom \( H_t \) is created in the procedure, then either it is in \( \Lambda_s \) or \( H_t \) is a new labeled vertex, which shows that \( k(V^*, \Lambda) \) will increase. Thus, the value of \( k(V^*, \Lambda) \) increases by at least one between two consecutive dual updates. Since the range of \( k(V^*, \Lambda) \) is at most \( O(n) \), the dual variables are updated at most \( O(n) \) times.

**10. AUGMENTATION**

When Search finds an augmenting path \( P \), we update the primal solution using \( P \). The augmentation procedure primarily replaces \( B^* \) with \( B^* \cup P \). In addition, it updates the bud, the tip, and its mate carefully. After the augmentation, the algorithm applies Search in each blossom \( H \) to obtain a new routing and ordering in \( H \).

In this proceedings version, we will only describe the outline of the augmentation procedure. See the full version [14] for details.

Suppose we are given \( V^*, B^*, C^*, \Lambda, b_i, t_i, t_i, p, q \). Let \( P \) be an augmenting path. In the augmentation along \( P \), we update \( V^*, B^*, C^*, \Lambda, b_i, t_i, t_i, p, q \). The new objects are denoted
by \( V^*, B^*, C^*, \lambda, b_i, t_i, \tilde{i}_i, p : V^* \to \mathbb{R} \), and \( q : \lambda \to \mathbb{R}_+ \), respectively. The procedure for augmentation is described as follows.

We first remove each blossom \( H_i \in \Lambda \) with \( q(H_i) = 0 \). Note that we also remove \( b_i, t_i, \) and \( \tilde{i}_i \) if \( H_i \in \Lambda_n \), and update \( V^*, B^*, C^*, \lambda, p, \) and \( q \), accordingly.

Let \( I_P := \{ i \in \{ 1, \ldots, \lambda \} \mid P \cap H_i \neq \emptyset \} \), and introduce three new vertices \( \tilde{b}_i, \tilde{t}_i, \) and \( \tilde{i}_i \) for each \( i \in I_P \), which will be a bud, a tip, and the mate of \( \tilde{i}_i \) after the augmentation. By the definition of augmenting paths, for each \( i \in I_P \), there exists a vertex \( x_i \in H_i^P \) such that \( P \cap H_i = R_{H_i}(x_i) \).

Define
\[
V' := V^* \cup \{ \tilde{b}_i, \tilde{t}_i, \tilde{i}_i \mid i \in I_P \},
\]
\[
B' := B^* \cup \{ \tilde{b}_i, \tilde{t}_i \mid i \in I_P, x_i \in B^* \}
\cup \{ \tilde{i}_i \mid i \in I_P, x_i \in V^* \setminus B^* \},
\]
\[
H_i' := H_i \cup \{ \tilde{i}_i \mid j \in I_P, H_j \subseteq H_i \}
\cup \{ \tilde{t}_i \mid j \in I_P, H_j \not\subseteq H_i \} \quad (i = 1, \ldots, \lambda),
\]
\[
\Lambda' := (H_1', \ldots, H_{\lambda}', \Lambda'),
\]
and define \( q' : \Lambda' \to \mathbb{R}_+ \) as \( q'(H_i') = q(H_i) \) for \( i = 1, \ldots, \lambda \). Let \( \Lambda'_n \) (resp. \( \Lambda'_n \)) be the set of blossoms in \( \Lambda' \) without (resp. with) a source line. Define \( \tilde{b}_i := b_i, \tilde{t}_i := t_i, \) and \( \tilde{i}_i := i \) for \( i \in \{ 1, \ldots, \lambda \} \setminus I_P \) with \( H_i' \in \Lambda'_n \). For \( H_i' \in \Lambda'_n \), regard \( \tilde{b}_i, \tilde{t}_i, \) and \( \tilde{i}_i \) as the bud of \( \tilde{H}_i' \), the tip of \( \tilde{H}_i', \) and the mate of \( \tilde{i}_i \), respectively.

Let \( I_P' := \{ i \in \{ 1, \ldots, \lambda \} \mid P \cap H_i \neq \emptyset, H_i \in \Lambda_n \} \) and remove \( \{ b_i, t_i, \tilde{i}_i \mid i \in I_P' \} \) from each object. More precisely, let \( V^* := V^* \setminus \{ b_i, t_i, \tilde{i}_i \mid i \in I_P' \} \) and \( B^* := B^* \setminus \{ b_i, t_i, \tilde{i}_i \mid i \in I_P' \} \). Let \( \tilde{H}_j := \tilde{H}_j' \setminus \{ b_i, t_i, \tilde{i}_i \mid i \in I_P' \} \) for each \( j \) and let \( \tilde{\Lambda} := (\tilde{H}_1, \ldots, \tilde{H}_\lambda) \).

Let \( \Lambda_n \) (resp. \( \Lambda_\emptyset \)) be the set of blossoms in \( \Lambda \) without (resp. with) a source line. Define \( \tilde{C}^* := \tilde{C}^*[V^*] \). Let \( \tilde{p} \) be the restriction of \( \tilde{p} \) to \( V^* \) and define \( q : \Lambda \to \mathbb{R}_+ \) by \( q(H_i) = q'(H_i') \) for each \( i \). For \( \tilde{H}_i \in \tilde{\Lambda} \), the bud of \( \tilde{H}_i \), the tip of \( \tilde{H}_i \), and the mate of \( \tilde{i}_i \) are \( \tilde{b}_i, \tilde{t}_i, \) and \( \tilde{i}_i \), respectively.

### 11 ALGORITHM DESCRIPTION AND COMPLEXITY

Our algorithm for the minimum-weight parity base problem is described as follows.

**Algorithm Minimum-Weight Parity Base**

**Step 1:** Split the weight \( w_{uv} \) into \( p(u) \) and \( p(v) \) for each line \( (u, v) \in L \), \( i.e., p(u) + p(v) = w_{uv} \). Execute the greedy algorithm for finding a base \( B \subseteq B \) with minimum value of \( p(B) = \sum_{u \in B} p(u) \). Set \( \Lambda = \emptyset \).

**Step 2:** If there is no source line, then return \( B := B^* \cap V \) as an optimal solution. Otherwise, apply Search. If Search returns \( \emptyset \), then go to Step 3. If Search finds an augmenting path, then go to Step 4.

**Step 3:** Update the dual variables as in Section 9. If \( \varepsilon = +\infty \), then conclude that there exists no parity base and terminate the algorithm. Otherwise, delete all blossoms \( H_i \) with \( q(H_i) = 0 \) from \( \Lambda \) and go to Step 2.

**Step 4:** Apply the augmentation procedure as in Section 10 (see [14] for details) and go back to Step 2.

We now analyze the complexity. Since \( |V^*| = O(n) \), an execution of the procedure Search as well as the dual update requires \( O(n^2) \) arithmetic operations. By Lemma 9.3, Step 3 is executed at most \( O(n) \) times per augmentation. In Step 4, Search and dual update are executed \( O(n) \) times. Thus, Search and dual update are executed \( O(n) \) times per augmentation, which requires \( O(n^2) \) operations. We note that it also requires \( O(n^2) \) operations to update \( C^* \) and \( G^* \) after augmentation. Since each augmentation reduces the number of source lines by two, the number of augmentations during the algorithm is \( O(m) \), where \( m = \text{rank } A \), and hence the total number of arithmetic operations is \( O(n^2m) \).

**Theorem 11.1.** Algorithm Minimum-Weight Parity Base finds a parity base of minimum weight or detects infeasibility with \( O(n^2m) \) arithmetic operations over \( K \).

If \( K \) is a finite field of fixed order, each arithmetic operation can be executed in \( O(1) \) time. Hence Theorem 11.1 implies the following.

**Corollary 11.2.** The minimum-weight parity base problem over an arbitrary finite field \( K \) can be solved in strongly polynomial time.
When \( K = \mathbb{Q} \), it is not obvious that a direct application of our algorithm runs in polynomial time. This is because we do not know how to bound the number of bits required to represent the entries of \( C^T \). However, the minimum-weight parity base problem over \( \mathbb{Q} \) can be solved in polynomial time by applying our algorithm over a sequence of finite fields.

**Theorem 11.3.** The minimum-weight parity base problem over \( \mathbb{Q} \) can be solved in time polynomial in the binary encoding length \( (A) \) of the matrix representation \( A \).

**Proof.** By multiplying each entry of \( A \) by the product of the denominators of all entries, we may assume that each entry of \( A \) is an integer. Let \( y \) be the maximum absolute value of the entries of \( A \), and put \( N := [m \log(ny)] \). Note that \( N \) is bounded by a polynomial in \( (A) \). We compute the \( N \) smallest prime numbers \( p_1, \ldots, p_N \). Since it is known that \( p_N \approx O(N \log N) \) by the prime number theorem, they can be computed in polynomial time by the sieve of Eratosthenes.

For \( i = 1, \ldots, N \), we consider the minimum-weight parity base problem over \( GF(p_i) \) where each entry of \( A \) is regarded as an element of \( GF(p_i) \). In other words, we consider the problem in which each operation is executed modulo \( p_i \). Since each arithmetic operation over \( GF(p_i) \) can be executed in polynomial time, we can solve the minimum-weight parity base problem over \( GF(p_i) \) in polynomial time by Theorem 11.1. Among all optimal solutions of these problems, the algorithm returns the best one \( B \). That is, \( B \) is the minimum weight parity set subject to \( |B| = m \) and \( det A[U, B] \equiv 0 \pmod{p_i} \) for some \( i \in \{1, \ldots, N\} \).

To see the correctness of this algorithm, we evaluate the absolute value of the subdeterminant of \( A \). For any subset \( X \subseteq V \) with \( |X| = m \), we have

\[
|det A[X]| \leq m^r 2^N \leq 2^N \prod_{i=1}^{N} p_i.
\]

This shows that \( det A[X] = 0 \) if and only if \( det A[U, B] \equiv 0 \pmod{p_i} \). Therefore, \( det A[U, X] \equiv 0 \pmod{p_i} \) for some \( i \in \{1, \ldots, N\} \), which shows that the output \( B \) is an optimal solution.

**ACKNOWLEDGEMENTS**

The authors thank Jim Geelen, Gyula Pap and Kenjiro Takazawa for fruitful discussions on the topic of this paper. We also thank the anonymous reviewers for their helpful comments. This work is supported by JST through CREST, No. JPMJCR14D2, and ERATO, No. JPMJER1305, and by Grants-in-Aid for Scientific Research No. JP24106002 and No. JP24106005 from MEXT, and No. JP16K16910 from JSPS.

**REFERENCES**

[1] J. Byrka, F. Grandoni, T. Rothvoß, L. Sanitá: Steiner tree approximation via iterative randomized rounding, J. ACM, 60 (2013), 6: 1–33.

[2] P. M. Camerini, G. Gability, F. Maffioli: Random pseudo-polynomial algorithms for exact matroid problems, J. Algorithms, 13 (1992), 258–273.