Electronic Preprint for Journal of Information Processing Vol.25

Regular Paper

An Improved-time Polynomial-space Exact Algorithm for TSP in Degree-5 Graphs

Norhazwani Md Yunos[1,a)] Aleksandar Shurbevski[2,b)] Hiroshi Nagamochi[2,c)]

Received: November 7, 2016, Accepted: May 16, 2017

Abstract: The Traveling Salesman Problem (TSP) is one of the most well-known NP-hard optimization problems. Following a recent trend of research which focuses on developing algorithms for special types of TSP instances, namely graphs of limited degree, in an attempt to reduce a part of the time and space complexity, we present a polynomial-space branching algorithm for the TSP in an n-vertex graph with degree at most 5, and show that it has a running time of $O^*(2.3500^n)$, which improves the previous best known time bound of $O^*(2.4723^n)$ given by the authors (the 12^{th} International Symposium on Operations Research and Its Application (ISORA 2015), pp.45–58, 2015). While the base of the exponent in the running time bound of our algorithm is greater than 2, it still outperforms Gurevich and Shelah's $O^*(4^n n^{\log n})$ polynomial-space exact algorithm for the TSP in general graphs (SIAM Journal of Computation, Vol.16, No.3, pp.486–502, 1987). In the analysis of the running time, we use the measure-and-conquer method, and we develop a set of branching rules which foster the analysis of the running time.

Keywords: Traveling Salesman Problem, exact exponential algorithm, branch-and-reduce, measure-and-conquer

1. Introduction

The Traveling Salesman Problem (TSP) is one of the most extensively studied problems across many fields of optimization. Briefly, given a set of cities and their pairwise distances, the TSP asks for a shortest closed route which visits each of the cities exactly once. It has been formulated as a mathematical problem in the 1930s, and many algorithmic methods have been investigated to address the challenge of finding the fastest algorithm in terms of running time. On the other hand, it has proven even more challenging to devise reasonable algorithms that would use a manageable amount of computation space, bounded by a polynomial in an input instance's size. We review previous algorithmic attempts, making a distinction between those which require space exponential in the size of a problem instance, and those requiring space polynomial in the input size. We use the O^* notation, which suppresses polynomial factors.

The first non-trivial algorithm for the TSP in an n-vertex graph is the $O^*(2^n)$-time dynamic programming algorithm discovered independently by Bellman [2], and Held and Karp [10] in the early 1960s. This dynamic programming algorithm however, requires also an exponential amount of space. Ever since, this running time has only been improved for special types of graphs. Primarily, investigation efforts have been focused on graphs in which vertices have a limited degree. Henceforth, let degree-i graph stand for a graph in which vertices have a maximum degree at most i. A recent improvement of the time bound to $O^*(1.2186^n)$ for degree-3 graphs has been presented by Bodlaender et al. [3], where the authors make use of a general approach for speeding up straightforward dynamic programming algorithms. For the TSP in degree-4 graphs, Gebauer [8] has shown a time bound of $O^*(1.733^n)$, by using a dynamic programming approach. The running space of these algorithms remains exponential in the size of an input instance.

In the vein of polynomial space algorithms, Gurevich and Shelah [9] have shown that the TSP in a general n-vertex graph is solvable in time $O^*(4^n n^{\log n})$. Eppstein [5] started the exploration into polynomial space TSP algorithms specialized for graphs of bounded degree by designing an algorithm for degree-3 graphs that runs in $O^*(1.260^n)$ time. He introduced a branch-and-search method by considering a generalization of the TSP proposed by Rubin [14], called the *forced* TSP. Iwama and Nakashima [11] claimed an improvement of Eppstein's time bound to $O^*(1.251^n)$ time for the TSP in degree-3 graphs. Later, Liskiewicz and Schuster [12] uncovered some oversights made in Iwama and Nakashima's analysis, and proved that their algorithm actually runs in $O^*(1.257^n)$ time. Liskiewicz and Schuster then made some minor modifications to Eppstein's algorithm and showed that this modified algorithm runs in $O^*(1.2553^n)$ time, a slight improvement over Iwama and Nakashima's algorithm. Xiao and Nagamochi [15] have recently presented an $O^*(1.2312^n)$-time algorithm for the TSP in degree-3 graphs, and this improved all previous time bounds for polynomial-space algorithms. They used the basic steps of Eppstein's branch-and-search algorithm, and introduced a branching rule based on a cut-circuit structure.

[1] Technical University of Malaysia Malacca, Malacca 76900, Malaysia
[2] Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Sakyo, Kyoto 606–8501, Japan
[a)] wanie.my@utem.edu.my
[b)] shurbevski@amp.i.kyoto-u.ac.jp
[c)] nag@amp.i.kyoto-u.ac.jp
This work has been done while the first author was a Ph.D. student at the Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University.

In the process of improving the time bound, they used a simple measure-and-conquer analysis, and effectively analyzed their algorithm by introducing an amortization scheme over the cut-circuit structure, setting weights to both vertices and connected components of induced graphs.

For the TSP in degree-4 graphs, Eppstein [5] designed an algorithm that runs in $O^*(1.890^n)$ time, based on a branch-and-search method. Following, Xiao and Nagamochi [16] showed an improved value for the upper bound of the running time and showed that their algorithm runs in $O^*(1.692^n)$ time. To the best of our knowledge, this is currently the fastest algorithm for the TSP in degree-4 graphs. Basically, the idea behind their algorithm is to apply reduction rules until no further reduction is possible, and then branch on an edge by either including it in a solution or excluding it from any solution. This is similar to most of the previous branch-and-search algorithms for the TSP. To effectively analyze their algorithm, Xiao and Nagamochi used the measure-and-conquer method by setting a weight to each vertex in an input graph. From each branching operation, they derived a branching vector using the assigned weight and evaluated how much weight can be decreased in each of the two instances obtained by branching on a selected edge e. In this way, they were able to analyze by how much the total weight decreases in each branch. Moreover, they indicated that the measure will decrease more if we select a "good" edge to branch on, and gave a set of simple rules based on a graph's topological properties for choosing such an edge. However, the analysis of the running time itself is not as straightforward, and the interested reader is referred to the original paper [16].

Md Yunos et al. [13] presented the first algorithm specialized for the TSP in degree-5 graphs, and showed that their algorithm runs in $O^*(2.4723^n)$ time. This algorithm employs similar techniques as most of the previous branching algorithms for the TSP. In the analysis, the authors use the measure-and-conquer method as a tool to get an upper bound of the running time. In this work we present an extended algorithm coupled with an improved analysis, giving an upper bound on the running time of $O^*(2.3500^n)$ for the TSP in degree-5 graphs with n vertices, which improves the previous result by the authors [13]. Namely, a new branching rule has been introduced for convenience in the analysis, and we present a much more detailed analysis, through which we are able to obtain an improved bound on the running time.

The remainder of this paper is organized as follows. Section 2 introduces the basic notation used in this paper and presents an introduction to branching algorithms and the measure-and-conquer method. Section 3 describes our polynomial-space branching algorithm. We state our main result in Section 4, where we proceed with the analysis of the proposed algorithm. Finally, Section 5 concludes the paper.

2. Methods

2.1 Preliminaries

For a graph G, let $V(G)$ denote the set of vertices in G, and let $E(G)$ denote the set of edges in G. A vertex u is a neighbor of a vertex v if u and v are adjacent by an edge uv. We denote the set of all neighbors of a vertex v by $N(v)$, also called the *neighborhood* of v, and denote by $d(v)$ the cardinality $|N(v)|$ of $N(v)$, also called the *degree* of v.

For a vertex v and subsets $W \subseteq V(G)$ of vertices, and $E' \subseteq E(G)$ of edges, we consider the following sets of vertices:

- $N(v; W) \triangleq N(v) \cap W$,
- $N_{E'}(v) \triangleq N(v) \cap \{u \mid uv \in E'\}$, and
- $N_{E'}(v; W) \triangleq N_{E'}(v) \cap W$.

Let $d_{E'}(v)$ and $d_{E'}(v, W)$ denote the cardinalities $|N_{E'}(v)|$ and $|N_{E'}(v, W)|$ of $N_{E'}(v)$ and $N_{E'}(v, W)$, respectively. Also, for a subset E' of $E(G)$, we denote by $G - E'$ the graph $(V, E \setminus E')$ obtained from G by removing the edges in E'.

We employ a known generalization of the TSP proposed by Rubin [14], named the *forced* Traveling Salesman Problem by Eppstein [5]. We define an instance $I = (G, F)$ that consists of a simple, edge weighted, undirected graph G, and a subset F of edges in G, called *forced*. For brevity, throughout this paper let U denote $E(G) \setminus F$. A vertex is called *forced* if exactly one of its incident edges is forced. Similarly, it is called *unforced* if no forced edge is incident to it. A Hamiltonian cycle in G is called a *tour* if it passes through all the forced edges in F. Under these circumstances, the forced TSP requests to find a minimum cost tour of an instance (G, F), or determine that none exists.

Throughout this paper, we assume that the maximum degree of a vertex in G is at most 5. We denote a forced (resp., unforced) vertex of degree i as a type fi vertex (resp., ui vertex). We are interested in six types of vertices in an instance of (G, F), namely, u5, f5, u4, f4, u3 and f3-vertices. Let $V_{\mathrm{f}i}$ (resp., $V_{\mathrm{u}i}$), $i = 3, 4, 5$, denote the set of fi-vertices (resp., ui-vertices) in (G, F).

2.2 A Polynomial-Space Branching Algorithm

Our algorithm contains two major steps which are repeated recursively. In the first step, the algorithm applies reduction rules until no further reduction is possible. Once a feasible instance which cannot be further reduced has been obtained, the algorithm branches in search of a solution by either including a chosen unforced edge e into F, **force**(e), or excluding it from $E(G)$, **delete**(e). The choice of the edge e to branch on is based on a set of *branching rules*. By applying a branching operation, the algorithm generates two new instances, called branches.

2.2.1 Reduction Rules

Reduction is the process of transforming an instance to a smaller instance while preserving its optimality. It takes polynomial time to obtain a solution of an original instance from a solution of a smaller instance that has been obtained by a reduction procedure from the original instance.

Not all instances of the forced TSP have a tour. If an instance has no tour, we call it *infeasible*. Observation 1 gives two sufficient conditions for an instance to be infeasible, as observed by Rubin [14].

Observation 1 An instance (G, F) is infeasible if one of the following conditions holds.

(i) $d(v) \leq 1$ for some vertex $v \in V(G)$, and
(ii) $d_F(v) \geq 3$ for some vertex $v \in V(G)$.

In this paper, there are two reduction rules which are applied following any branching operation, following Observation 2.

Observation 2 Each of the following reductions preserves

the feasibility and a minimum cost tour of an instance (G, F).

(i) If $d(v) = 2$ for a vertex v, then add to F any unforced edge incident to the vertex v; and

(ii) If $d(v) > 2$ and $d_F(v) = 2$ for a vertex v, then remove from G any unforced edge incident to the vertex v.

An instance (G, F) which does not satisfy any of the conditions in Observation 1 and Observation 2 is called a *reduced* instance.

Based on Observation 1 and Observation 2, we can form a reduction procedure that given an instance (G, F) of the forced TSP returns a reduced instance, or a message for the infeasibility of (G, F), which for convenience we can assume that evaluates to ∞. For completeness, we describe such a procedure as Algorithm Red(G, F).

Algorithm Red(G, F)

Input: An instance (G, F).
Output: A reduced instance (G', F') of (G, F); or a message for the infeasibility of (G, F) which evaluates to ∞.

Initialize $(G', F') := (G, F)$;
while (G', F') is not a reduced instance **do**
  if there is a vertex v in (G', F') such that $d(v) \leq 1$ or $d_{F'}(v) \geq 3$ **then**
    return ∞
  else if there is a vertex v in (G', F') such that $2 = d(v) > d_{F'}(v)$ **then**
    Let $E^\dagger$ be the set of unforced edges incident to all such vertices;
    set $F' := F' \cup E^\dagger$
  else if there is a vertex v in (G', F') such that $d(v) > d_{F'}(v) = 2$ **then**
    Let $E^\dagger$ be the set of unforced edges incident to all such vertices;
    set $G' := G' - E^\dagger$
  end if
end while;
return (G', F')

2.2.2 Branching Rules

To describe our branching algorithm, let (G, F) be a reduced instance such that the maximum degree of G is at most 5. In (G, F), an unforced edge $e = vt$ incident to a vertex v of degree 5 is called *optimal*, if it satisfies a condition c-i below with minimum index i, over all unforced edges vt in (G, F). We refer to the following conditions for choosing an optimal edge to branch on, c-1 to c-15, as the branching rules.

(c-1) $v \in V_{\mathrm{f5}}$ and $t \in N_U(v; V_{\mathrm{f3}})$ such that $N_U(v) \cap N_U(t) = \emptyset$;
(c-2) $v \in V_{\mathrm{f5}}$ and $t \in N_U(v; V_{\mathrm{f3}})$ such that $N_U(v) \cap N_U(t) \neq \emptyset$;
(c-3) $v \in V_{\mathrm{f5}}$ and $t \in N_U(v; V_{\mathrm{u3}})$;
(c-4) $v \in V_{\mathrm{f5}}$ and $t \in N_U(v; V_{\mathrm{f4}})$ such that $N_U(v) \cap N_U(t) = \emptyset$;
(c-5) $v \in V_{\mathrm{f5}}$ and $t \in N_U(v; V_{\mathrm{f4}})$ such that $N_U(v) \cap N_U(t) \neq \emptyset$;
  (I) $|N_U(v) \cap N_U(t)| = 1$; and
  (II) $|N_U(v) \cap N_U(t)| = 2$;
(c-6) $v \in V_{\mathrm{f5}}$ and $t \in N_U(v; V_{\mathrm{u4}})$;
(c-7) $v \in V_{\mathrm{f5}}$ and $t \in N_U(v; V_{\mathrm{f5}})$ such that $N_U(v) \cap N_U(t) = \emptyset$;
(c-8) $v \in V_{\mathrm{f5}}$ and $t \in N_U(v; V_{\mathrm{f5}})$ such that $N_U(v) \cap N_U(t) \neq \emptyset$;
  (I) $|N_U(v) \cap N_U(t)| = 1$;
  (II) $|N_U(v) \cap N_U(t)| = 2$; and
  (III) $|N_U(v) \cap N_U(t)| = 3$;
(c-9) $v \in V_{\mathrm{f5}}$ and $t \in N_U(v; V_{\mathrm{u5}})$;
(c-10) $v \in V_{\mathrm{u5}}$ and $t \in N_U(v; V_{\mathrm{f3}})$ such that $N_U(v) \cap N_U(t) \neq \emptyset$;
  Assume without loss of generality that $N_U(v) \cap N_U(t) = \{t_2\}$.
  (I) t_2 is an f3-vertex,
  (II) t_2 is a u3-vertex,
  (III) t_2 is an f4-vertex,
  (IV) t_2 is a u4-vertex, and
  (V) t_2 is a u5-vertex.
(c-11) $v \in V_{\mathrm{u5}}$ and $t \in N_U(v; V_{\mathrm{f3}})$, such that $N_U(v) \cap N_U(t) = \emptyset$;
(c-12) $v \in V_{\mathrm{u5}}$ and $t \in N_U(v; V_{\mathrm{u3}})$;
(c-13) $v \in V_{\mathrm{u5}}$ and $t \in N_U(v; V_{\mathrm{f4}})$;
(c-14) $v \in V_{\mathrm{u5}}$ and $t \in N_U(v; V_{\mathrm{u4}})$; and
(c-15) $v \in V_{\mathrm{u5}}$ and $t \in N_U(v; V_{\mathrm{u5}})$.

The collective set of branching rules are illustrated in **Fig. 1**. The given list of branching rules is exhaustive, in the sense that each possible combination of a degree 5 vertex and choice for its neighbor are included.

For convenience in the analysis of the algorithm, cases c-5 and c-8 have been subdivided into sub-cases according to the cardinality of the intersection of the neighborhoods of the vertices v and t. Intersections of lower cardinality take precedence over higher ones. Case c-10 has been subdivided into sub-cases according to the type of the unique vertex in the neighborhood intersection of vertices v and t.

Given a reduced instance (G, F), our algorithm checks whether there exists a vertex of degree 5. If it does, then the algorithm chooses an optimal edge according to the branching rules, and branches into two recursive branches; one branch that includes the selected edge into the set of forced edges, and the other branch removes the edge from the set of edges of the graph G. If there does not exist a vertex of degree 5 in the graph G of the given instance, then we can call a polynomial space exact algorithm for the TSP that is specialized for degree-4 graphs, e.g., the algorithm due to Xiao and Nagamochi [16]. Our branching algorithm is described in Algorithm tsp5(G, F).

Algorithm tsp5(G, F)

Input: An instance (G, F) such that the maximum degree of G is at most 5.
Output: The minimum cost of a tour of (G, F); or a message for the infeasibility of (G, F) which evaluates to ∞.

Run Red(G, F);
if Red(G, F) returns ∞ **then**
  return ∞
else
  Let $(G', F') :=$ Red(G, F);
  if $V_{\mathrm{u5}} \cup V_{\mathrm{f5}} \neq \emptyset$ in (G', F') **then**
    Choose an optimal unforced edge e;
    return min{tsp5$(G', F' \cup \{e\})$, tsp5$(G' - \{e\},\ F')$}
  else /* the maximum degree of any vertex in (G', F') is at most 4 */
    return tsp4(G', F').
  end if
end if.

Note: The input and output of algorithm tsp4(G, F) are as follows:
Input: An instance (G, F) such that the maximum degree of G is at most 4.
Output: The minimum cost of a tour of (G, F); or a message for the infeasibility of (G, F) which evaluates to ∞.

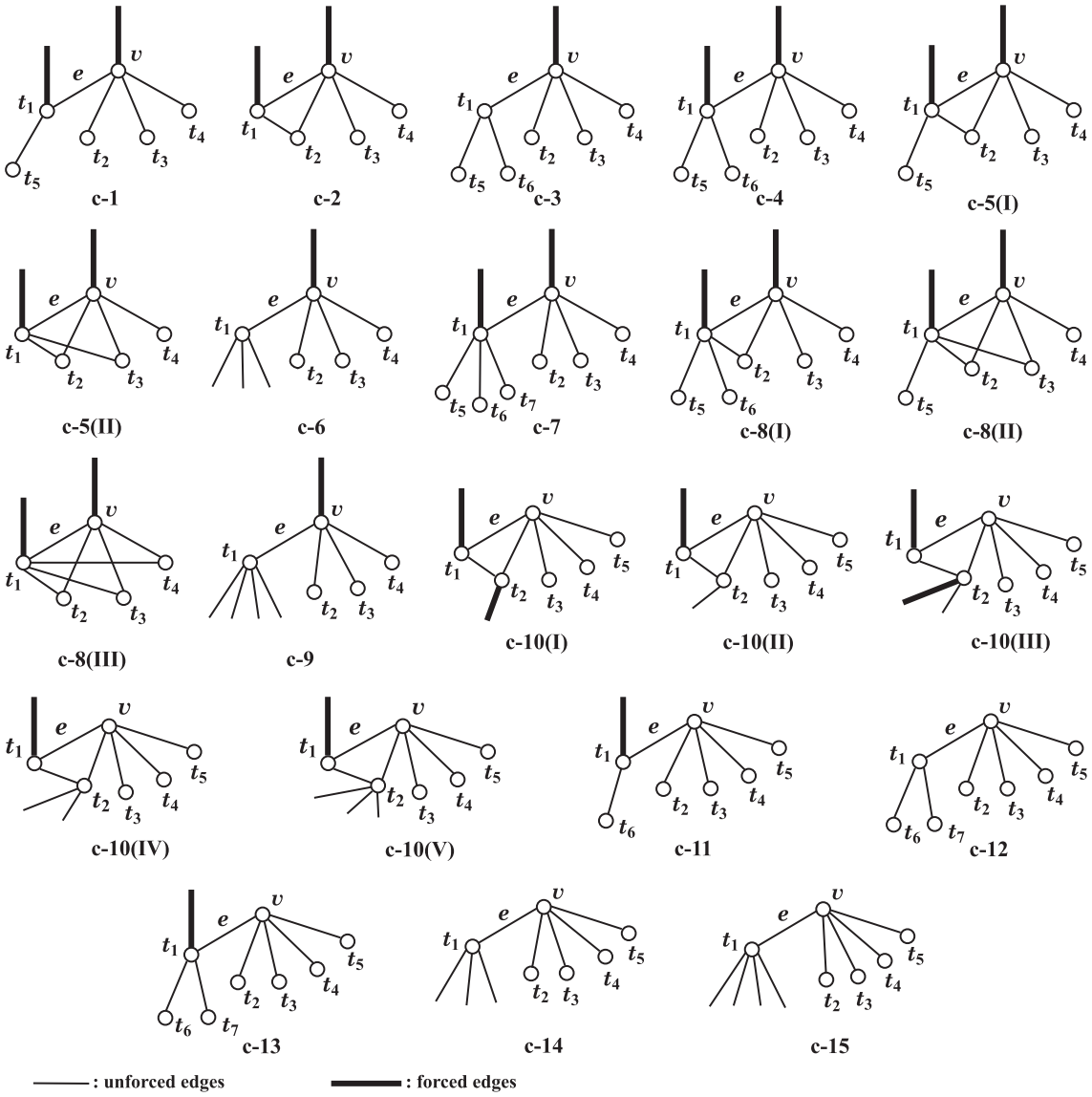

Fig. 1 Illustration of the Branching Rules.

3. Analysis

3.1 Analysis Framework

To effectively analyze the running time of our branching algorithm, we use the measure-and-conquer method as introduced by Fomin et al. [6]. Given an instance $I = (G, F)$ of the forced TSP, we assign a nonnegative weight $\omega(v)$ to each vertex $v \in V(G)$ according to its type. To this effect, we set a non-negative vertex weight function $\omega : V \to \mathbb{R}_+$ in the graph G, and we use the sum of weights of all vertices in the graph as the measure $\mu(I)$ of instance I, that is,

$$\mu(I) \triangleq \sum_{v \in V(G)} \omega(v). \tag{1}$$

It is important for the analysis to find a measure which satisfies the following properties:

(i) $\mu(I) = 0$ if and only if I can be solved in polynomial time; and

(ii) If I' is a sub-instance of I obtained through a reduction or a branching operation, then $\mu(I') \le \mu(I)$.

We call a measure μ satisfying conditions (i) and (ii) above a *proper measure*.

A branching algorithm typically comprises multiple branching rules. We perform the time analysis of the branching algorithm via appropriately constructed recurrences over the measure $\mu = \mu(I)$ of an instance $I = (G, F)$, for each branching rule of the algorithm. Let $T(\mu)$ denote the number of nodes in the search tree generated by a branching rule when invoked on the instance I with measure μ. For $k \ge 2$, let $I^1, I^2, \ldots, I^k$ be the instances obtained from I by the branching operation, and for $i = 1, 2, \ldots, k$, let $t_i \le \mu(I) - \mu(I^i)$ be lower bounds on the amounts of decrease in the measure. We call $(t_1, t_2, \ldots, t_k)$ a *branching vector* of the branching operation, and this implies the linear recurrence

$$T(\mu) \le T(\mu - t_1) + T(\mu - t_2) + \cdots + T(\mu - t_k). \tag{2}$$

To evaluate the performance of this branching vector, we can use any standard method for linear recurrence relations. In fact, it is known that $T(\mu)$ is of the form $O(\tau^\mu)$, where τ is the unique positive real root of the function $f(x) = 1 - (x^{-t_1} + x^{-t_2} + \cdots + x^{-t_k})$. The value τ is called the *branching factor* of the branching vector

$(t_1, t_2, \ldots, t_k)$. The running time of the algorithm is determined by considering the worst branching factor over all branching vectors generated by all of the branching rules of the algorithm.

For further details justifying this approach, as well as a solid introduction to branching algorithms, the reader is referred to the book of Fomin and Kratsch [7].

3.2 Weight Setting

In order to obtain a measure which will naturally give a running time bound as a function of the size of a TSP instance, we require that the weight of each vertex be at most 1. In what follows, we examine some necessary constraints which the vertex weights should satisfy in order for us to obtain a proper measure.

For each $i \in \{3, 4, 5\}$, we denote by w_i the weight of a ui-vertex, and by w'_i the weight of an fi-vertex. The conditions for a proper measure require that the measure of an instance obtained through a reduction operation or a branching operation be not greater than the measure of the original instance. Thus, the vertex weights should satisfy the following relations:

$$w_5 \leq 1, \tag{3}$$

$$w'_i \leq w_i, \quad i = 3, 4, 5, \tag{4}$$

$$w_3 \leq w_4 \leq w_5, \text{ and} \tag{5}$$

$$w'_3 \leq w'_4 \leq w'_5. \tag{6}$$

The vertex weight for vertices of degree less than 3 is set to be 0.

Lemma 1 states that given Algorithm Red(G, F) and Algorithm tsp5(G, F), setting vertex weights which satisfy the conditions of Eqs. (4) to (6) is sufficient to obtain a proper measure. This lemma has been proved by Md Yunos et al. [13].

Lemma 1 If the weights of vertices are chosen as in Eqs. (4) to (6), then the measure $\mu(I)$ never increases as a result of the reduction or the branching operations of Algorithm Red(G, F) and Algorithm tsp5(G, F).

Proof. Let $I = (G, F)$ be a given instance of the forced TSP. Due to our definition of the measure $\mu(I)$ of Eq. (1), it suffices to show that none of the individual vertex weights will increase as a result of a reduction operation or a branching operation in Algorithm Red(G, F) and Algorithm tsp5(G, F).

The branching rules state that for an unforced edge e in $E(G) \setminus F$, two sub-instances are generated by either setting $F := F \cup \{e\}$, termed **force**(e), or by setting $G := G - \{e\}$, termed **delete**(e). We bring to the reader's attention that a reduction operation is in fact a repeated application of the above two steps, **force**(e) or **delete**(e), for some unforced edge e, identified by the conditions in Observation 2. Therefore, we proceed with analyzing the effects of applying each of the **force**(e) or the **delete**(e) operations.

Let $e = uv$ be an unforced edge to which one of the **force**(e) or **delete**(e) operations will be applied. Without loss of generality, we observe the effect of the operation on the vertex weight $\omega(v)$.

In the case where operation **force**(e) is applied, the following cases may arise.

- If v is an unforced vertex, then v will become forced. By Eq. (4), the weight $\omega(v)$ will not increase.
- If v is a forced vertex, then $\omega(v)$ will become 0; and
- If $d_F(v) \geq 2$, then by Observation 1 the instance will become infeasible.

On the other hand, if operation **delete**(e) is applied, then we observe the following cases.

- If v is either forced or unforced, and $d(v) \geq 3$, then the degree of v will decrease by one, and by Eqs. (5) and (6), $\omega(v)$ will not increase; and
- If v is either forced or unforced, and $d(v) \leq 2$, then by Observation 1 the instance will become infeasible.

Following the above observations, we conclude that the complete measure $\mu(I)$ of a given instance $I = (G, F)$ of the forced TSP will not increase as a result of the reduction and the branching operations in Algorithm Red(G, F) and Algorithm tsp5(G, F).

To simplify some arguments and the list of the branching vectors we are about to derive, we introduce the following notation:

$$\Delta_i \triangleq w_i - w'_i, \quad 3 \leq i \leq 5$$

$$\Delta_{i,j} \triangleq w_i - w_j, \quad 3 \leq j < i \leq 5, \text{ and}$$

$$\Delta'_{i,j} \triangleq w'_i - w'_j, \quad 3 \leq j < i \leq 5,$$

further,

$$m_1 \triangleq \min\{w'_3,\ w_3,\ \Delta'_{4,3},\ \Delta_{4,3},\ \Delta'_{5,4},\ \Delta_{5,4}\}, \tag{7}$$

$$m_2 \triangleq \min\{w_3,\ \Delta'_{4,3},\ \Delta_{4,3},\ \Delta'_{5,4},\ \Delta_{5,4}\}, \tag{8}$$

$$m_3 \triangleq \min\{w'_3,\ \Delta_3,\ w'_4,\ \Delta_4,\ w'_5,\ \Delta_5\}, \tag{9}$$

$$m_4 \triangleq \min\{\Delta'_{4,3},\ \Delta_{4,3},\ \Delta'_{5,4},\ \Delta_{5,4}\}, \tag{10}$$

$$m_5 \triangleq \min\{w'_4,\ w_4,\ \Delta'_{5,3},\ \Delta_{5,3}\}, \tag{11}$$

$$m_6 \triangleq \min\{\Delta_{4,3},\ \Delta'_{5,4},\ \Delta_{5,4}\}, \tag{12}$$

$$m_7 \triangleq \min\{\Delta'_{5,4},\ \Delta_{5,4}\}, \tag{13}$$

$$m_8 \triangleq \min\{\Delta'_{5,3},\ \Delta_{5,3}\}, \tag{14}$$

$$m_9 \triangleq \min\{w'_3,\ \Delta_3,\ w'_4,\ \Delta_4,\ \Delta_5\}, \tag{15}$$

$$m_{10} \triangleq \min\{w'_3,\ w_3,\ \Delta'_{4,3},\ \Delta_{4,3},\ \Delta_{5,4}\}, \text{ and} \tag{16}$$

$$m_{11} \triangleq \min\{w'_3, w_3, w'_4, w_4 - w'_3, w_5 - w'_4\}. \tag{17}$$

3.3 Main Results

We bring to the reader's attention the fact that the number n of vertices in the graph G remains unmodified throughout the process of the reduction and the branching operations. In addition to seeking a proper measure, we also require that the weight of each vertex be at most 1, and therefore, the measure $\mu(I)$ will not be greater than the number n of vertices in G. As a consequence, a running time bound as a function of the measure $\mu(I)$ implies the same running time bound as a function of n. The weight assigned to each vertex type plays an important role, since the value of the branching factor depends solely on these weights.

Let the vertex weight function $\omega(v)$ be chosen as follows:

$$\omega(v) \triangleq \begin{cases} w_5 = 1.000000 & \text{for a u5-vertex } v \\ w'_5 = 0.714884 & \text{for an f5-vertex } v \\ w_4 = 0.653197 & \text{for a u4-vertex } v \\ w'_4 = 0.368082 & \text{for an f4-vertex } v \\ w_3 = 0.306395 & \text{for a u3-vertex } v \\ w'_3 = 0.160872 & \text{for an f3-vertex } v \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

The vertex weight function $\omega(v)$ given in Eq. (18) is obtained

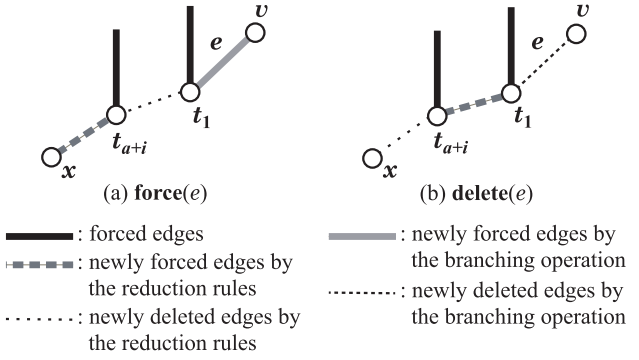

Fig. 2 Illustration of forced and deleted edges by the branching operation and reduction rules for an f3 vertex.

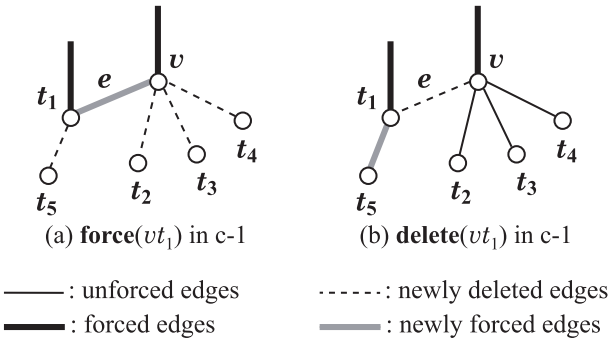

Fig. 3 Illustration of branching rule c-1, where vertex $v \in V_{\mathrm{f5}}$ and vertex $t_1 \in N_U(v; V_{\mathrm{f3}})$ such that $N_U(v) \cap N_U(t_1) = \emptyset$.

as a solution to a quasiconvex program, according to the method introduced by Eppstein [4]. All the branching vectors are in fact constraints in the quasiconvex program.

Lemma 2 If the vertex weight function $\omega(v)$ is set as in Eq. (18), then each branching operation in Algorithm tsp5(G, F) has a branching factor not greater than 2.349978.

A proof of Lemma 2 will be derived analytically in the several subsections which follow. From the lemma, we get our main result as stated in Theorem 1.

Theorem 1 The TSP in an n-vertex graph G with maximum degree 5 can be solved in $O^*(2.3500^n)$ time and polynomial space.

In the remainder of the analysis, for an optimal edge $e = vt_1$, we denote $N_U(v)$ by $\{t_1, t_2, \ldots, t_a\}$, $a = d_U(v)$, and $N_U(t_1) \setminus \{v\}$ by $\{t_{a+1}, t_{a+2}, \ldots, t_{a+b}\}$, $b = d_U(t_1) - 1$. If $N_U(v) \cap N_U(t_1) \neq \emptyset$, then let $c = |N_U(v) \cap N_U(t_1)|$, and we can assume without loss of generality that $t_{1+i} = t_{a+i}$ for $i = 1, 2, \ldots, c$.

If $N_U(v) \cap N_U(t_1) = \emptyset$, and there exists an f3-vertex t_{a+i} in $N_U(t_1) \setminus \{v\}$, let $x \in N_U(t_{a+i}) \setminus \{t_1\}$. We see that the choice of vertex x is unique, because t_{a+i} is of type f3 and $|N_U(t_{a+i}) \setminus \{t_1\}| = 1$. This vertex x plays a key role in our analysis, namely in cases c-1 and c-11, as shown in **Fig. 2**.

3.4 Branching on Edges Around f5-vertices (Branching Rules c-1 to c-9)

This section derives branching vectors for the branching operations on an optimal edge $e = vt_1$, incident to an f5-vertex v, distinguishing nine cases for conditions c-1 to c-9.

Case c-1. There exist vertices $v \in V_{\mathrm{f5}}$ and $t_1 \in N_U(v; V_{\mathrm{f3}})$ such that $N_U(v) \cap N_U(t_1) = \emptyset$ (see **Fig. 3**): We branch on edge vt_1. Note that $N_U(t_1) \setminus \{v\} = \{t_5\}$.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and edges vt_2, vt_3, vt_4 and t_1t_5 will be deleted from G' by the reduction rules. Both v and t_1 will become vertices of degree 2. From Eq. (18), the weight of vertices of degree 2 is 0. So, the weight of vertex v decreases by w'_5 and the weight of vertex t_1 decreases by w'_3.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation, and the edge t_1t_5 will be added to F' by the reduction rules. The weight of vertex v decreases by $\Delta'_{5,4}$ and the weight of vertex t_1 decreases by w'_3.

There are two cases for the vertex t_5; 1) vertex t_5 is of type f3, and 2) otherwise. We will analyze these two cases separately for each branch **force**(vt_1) and **delete**(vt_1).

First, we analyze the case where the vertex t_5 is an f3-vertex (see Fig. 2). Recall that in such a case we denote by x the unique vertex in $N_U(t_5) \setminus \{t_1\}$. There are two cases for the vertex x; a) vertex $x \notin N_U(v)$, and b) vertex $x \in N_U(v)$.

First, we analyze the case where the vertex $x \notin N_U(v)$. In the branch of **force**(vt_1), the edge xt_5 will be added to F' by the reduction rules. Hence, each of the vertices t_2, t_3 and t_4 must be one of types f3, u3, f4, u4, f5, and a u5-vertex, and each of their weights decreases by at least $m_1 = \min\{w'_3, w_3, \Delta'_{4,3}, \Delta_{4,3}, \Delta'_{5,4}, \Delta_{5,4}\}$. The weight of vertex t_5 decreases by w'_3. If the vertex x is an f3-vertex (resp., u3, f4, u4, f5, and a u5-vertex), then the weight decrease α_1 of vertex x will be w'_3 (resp., Δ_3, w'_4, Δ_4, w'_5, and Δ_5). Thus, the total weight decrease for this case in the branch of **force**(vt_1) is at least $w'_5 + w'_3 + 3m_1 + w'_3 + \alpha_1$.

In the branch of **delete**(vt_1), the edge xt_5 will be deleted from G' by the reduction rules. Hence, the weight of vertex t_5 decreases by w'_3. If the vertex x is an f3-vertex (resp., u3, f4, u4, f5, and a u5-vertex), then the weight decrease β_1 of vertex x will be w'_3 (resp., w_3, $\Delta'_{4,3}$, $\Delta_{4,3}$, $\Delta'_{5,4}$, and $\Delta_{5,4}$). Thus, the total weight decrease for this case in the branch of **delete**(vt_1) is at least $w'_5 - w'_4 + w'_3 + w'_3 + \beta_1$.

As a result, for the ordered pair (α_1, β_1) taking values in $\{(w'_3, w'_3), (\Delta_3, w_3), (w'_4, \Delta'_{4,3}), (\Delta_4, \Delta_{4,3}), (w'_5, \Delta'_{5,4}), (\Delta_5, \Delta_{5,4})\}$, we get the following six branching vectors:

$$(w'_5 + 2w'_3 + 3m_1 + \alpha_1,\ w'_5 - w'_4 + 2w'_3 + \beta_1). \tag{19}$$

Second, we analyze the case where the vertex $x \in N_U(v)$. Without loss of generality, we assume that x is t_2. In the branch of **force**(vt_1), the edge t_2t_5 will be added to F' by the reduction rules. Each of the vertices t_3 and t_4 must be one of types f3, u3, f4, u4, f5, and a u5-vertex, and each of their weights decreases by at least $m_1 = \min\{w'_3, w_3, \Delta'_{4,3}, \Delta_{4,3}, \Delta'_{5,4}, \Delta_{5,4}\}$. The weight of vertex t_5 decreases by w'_3. If the vertex t_2 is an f3-vertex (resp., u3, f4, u4, f5, and a u5-vertex), then the weight decrease α_2 of vertex x will be w'_3 (resp., w_3, w'_4, $w_4 - w'_3$, w'_5, and $w_5 - w'_4$). Thus, the total weight decrease for this case in the branch of **force**(vt_1) is at least $w'_5 + w'_3 + 2m_1 + w'_3 + \alpha_2$.

In the branch of **delete**(vt_1), the edge t_2t_5 will be deleted from G' by the reduction rules. Hence, the weight of vertex t_5 decreases by w'_3. If the vertex t_2 is an f3-vertex (resp., u3, f4, u4, f5, and a u5-vertex), then the weight decrease β_2 of vertex x will be w'_3 (resp., w_3, $\Delta'_{4,3}$, $\Delta_{4,3}$, $\Delta'_{5,4}$, and $\Delta_{5,4}$). Thus, the total weight decrease for this case in the branch of **delete**(vt_1) is at

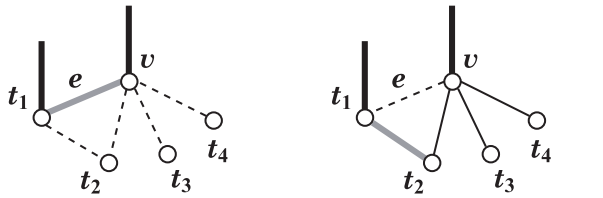

Fig. 4 Illustration of branching rule c-2, where vertex $v \in V_{\mathrm{f5}}$ and vertex $t_1 \in N_U(v; V_{\mathrm{f3}})$ such that $N_U(v) \cap N_U(t_1) = \{t_2\}$.

least $w'_5 - w'_4 + w'_3 + w'_3 + \beta_2$.

As a result, for the ordered pair (α_2, β_2) taking values in $\{(w'_3, w'_3),\ (w_3, w_3),\ (w'_4, \Delta'_{4,3}),\ (w_4 - w'_3, \Delta_{4,3}),\ (w'_5, \Delta'_{5,4}),\ (w_5 - w'_4, \Delta_{5,4})\}$, we get the following six branching vectors:

$$\left(w'_5 + 2w'_3 + 2m_1 + \alpha_2,\ w'_5 - w'_4 + 2w'_3 + \beta_2\right). \tag{20}$$

Next, we examine the case where the vertex t_5 is not an f3-vertex. In the branch of **force**(vt_1), if the vertex t_5 is a u3-vertex (resp., f4, u4, f5, and a u5-vertex), then the weight decrease α_3 of vertex t_5 will be w_3 (resp., $\Delta'_{4,3}$, $\Delta_{4,3}$, $\Delta'_{5,4}$, and $\Delta_{5,4}$). Thus, the total weight decrease for this case in the branch of **force**(vt_1) is at least $w'_5 + w'_3 + 3m_1 + \alpha_3$.

In the branch of **delete**(vt_1), if the vertex t_5 is a u3-vertex (resp., f4, u4, f5, and a u5-vertex), then the weight decrease β_3 of vertex t_5 will be Δ_3 (resp., w'_4, Δ_4, w'_5, and Δ_5). Thus, the total weight decrease for the case where t_5 is not an f3-vertex in the branch of **delete**(vt_1) is at least $w'_5 - w'_4 + w'_3 + \beta_3$.

As a result, for the ordered pair (α_3, β_3) taking values in $\{(w_3, \Delta_3),\ (\Delta'_{4,3}, w'_4),\ (\Delta_{4,3}, \Delta_4),\ (\Delta'_{5,4}, w'_5),\ (\Delta_{5,4}, \Delta_5)\}$, we get the following five branching vectors:

$$\left(w'_5 + w'_3 + 3m_1 + \alpha_3,\ w'_5 - w'_4 + w'_3 + \beta_3\right). \tag{21}$$

Case c-2. Case c-1 is not applicable, and there exist vertices $v \in V_{\mathrm{f5}}$ and $t_1 \in N_U(v; V_{\mathrm{f3}})$ such that $N_U(v) \cap N_U(t_1) = \{t_2\}$ (see **Fig. 4**): We branch on edge vt_1.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and edges vt_2, vt_3, vt_4 and t_1t_2 will be deleted from G' by the reduction rules. So, the weight of vertex v decreases by w'_5, and the weight of vertex t_1 decreases by w'_3. Each of the vertices t_3 and t_4 must be one of types f3, u3, f4, u4, f5, and a u5-vertex, and each of their weights decreases by at least $m_1 = \min\{w'_3, w_3, \Delta'_{4,3}, \Delta_{4,3}, \Delta'_{5,4}, \Delta_{5,4}\}$.

If the vertex t_2 is an f3 or a u3-vertex, after performing the branching operation, t_2 becomes a vertex of degree 1. From Observation 1, case (i), this is infeasible, and the algorithm will return ∞ and terminate. Otherwise, if the vertex t_2 is an f4-vertex (resp., u4, f5, and a u5-vertex), then the weight decrease α_4 of vertex t_2 will be w'_4 (resp., w_4, $\Delta'_{5,3}$, and $\Delta_{5,3}$). Thus, the total weight decrease in the branch of **force**(vt_1) is at least $w'_5 + w'_3 + 2m_1 + \alpha_4$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation, and the edge t_1t_2 will be added to F' by the reduction rules. So, the weights of vertices v and t_1 decrease by $\Delta'_{5,4}$ and w'_3, respectively. If the vertex t_2 is an f4-vertex (resp., u4, f5, and a u5-vertex), then the weight decrease β_4 of vertex t_2 will be w'_4 (resp., Δ_4, w'_5, and Δ_5). Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $w'_5 - w'_4 + w'_3 + \beta_4$.

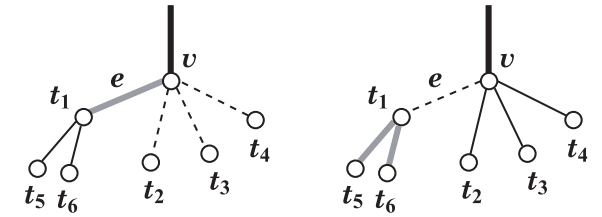

Fig. 5 Illustration of branching rule c-3, where vertex $v \in V_{\mathrm{f5}}$ and vertex $t_1 \in N_U(v; V_{\mathrm{u3}})$.

As a result, for the ordered pair (α_4, β_4) taking values in $\{(w'_4, w'_4),\ (w_4, \Delta_4),\ (\Delta'_{5,3}, w'_5),\ (\Delta_{5,3}, \Delta_5)\}$, we get the following four branching vectors:

$$(w'_5 + w'_3 + 2m_1 + \alpha_4,\ w'_5 - w'_4 + w'_3 + \beta_4). \tag{22}$$

Case c-3. Cases c-1 and c-2 are not applicable, and there exist vertices $v \in V_{\mathrm{f5}}$ and $t_1 \in N_U(v; V_{\mathrm{u3}})$ (see **Fig. 5**): We branch on edge vt_1. Note that $N_U(t_1) \setminus \{v\} = \{t_5, t_6\}$.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and edges vt_2, vt_3 and vt_4 will be deleted from G' by the reduction rules. So, the weights of vertices v and t_1 decrease by w'_5 and Δ_3, respectively. None of the vertices t_2, t_3 and t_4 can be an f3-vertex because it would have been chosen as an optimal edge in some previous case. Hence, each of the vertices t_2, t_3 and t_4 must be one of types u3, f4, u4, f5, and a u5-vertex, and each of their weights decreases by at least $m_2 = \min\{w_3, \Delta'_{4,3}, \Delta_{4,3}, \Delta'_{5,4}, \Delta_{5,4}\}$. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $w'_5 + w_3 - w'_3 + 3m_2$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation and edges t_1t_5 and t_1t_6 will be added to F' by the reduction rules. So, the weight of vertex v decreases by $\Delta'_{5,4}$ and the weight of vertex t_1 decreases by w_3. Each of the vertices t_5 and t_6 must be one of types f3, u3, f4, u4, f5, and a u5-vertex, and each of their weights decreases by at least $m_3 = \min\{w'_3, \Delta_3, w'_4, \Delta_4, w'_5, \Delta_5\}$. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $w'_5 - w'_4 + w_3 + 2m_3$.

As a result, we get the following branching vector:

$$(w'_5 + w_3 - w'_3 + 3m_2,\ w'_5 - w'_4 + w_3 + 2m_3). \tag{23}$$

Case c-4. None of the previous cases are applicable, and there exist vertices $v \in V_{\mathrm{f5}}$ and $t_1 \in N_U(v; V_{\mathrm{f4}})$ such that $N_U(v) \cap N_U(t_1) = \emptyset$ (see **Fig. 6**): We branch on edge vt_1. Note that $N_U(t_1) \setminus \{v\} = \{t_5, t_6\}$.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and edges vt_2, vt_3, vt_4, t_1t_5 and t_1t_6 will be deleted from G' by the reduction rules. So, the weight of vertex v decreases by w'_5 and the weight of vertex t_1 decreases by w'_4. Each of the vertices t_2, t_3 and t_4 must be one of types f4, u4, f5, and a u5-vertex, and each of their weights decreases by at least $m_4 = \min\{\Delta'_{4,3}, \Delta_{4,3}, \Delta'_{5,4}, \Delta_{5,4}\}$. Each of the vertices t_5 and t_6 must be one of types f3, u3, f4, u4, f5,

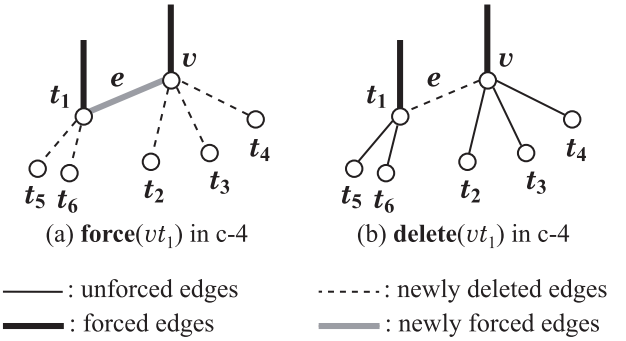

Fig. 6 Illustration of branching rule c-4, where vertex $v \in V_{\mathrm{f5}}$ and vertex $t_1 \in N_U(v; V_{\mathrm{f4}})$, such that $N_U(v) \cap N_U(t_1) = \emptyset$.

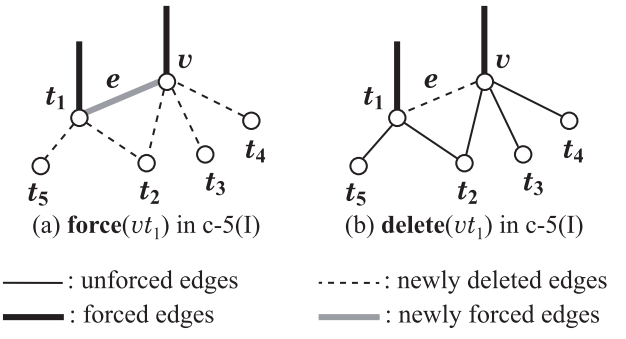

Fig. 7 Illustration of branching rule c-5(I), where vertex $v \in V_{\mathrm{f5}}$ and vertex $t_1 \in N_U(v; V_{\mathrm{f4}})$, such that $N_U(v) \cap N_U(t_1) = \{t_2\}$.

and a u5-vertex, and each of their weights decreases by at least $m_1 = \min\{w'_3, w_3, \Delta'_{4,3}, \Delta_{4,3}, \Delta'_{5,4}, \Delta_{5,4}\}$. Thus, the total decrease in the branch of **force**(vt_1) is at least $w'_5 + w'_4 + 3m_4 + 2m_1$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation. So, the weight of vertex v decreases by $\Delta'_{5,4}$ and the weight of vertex t_1 decreases by $\Delta'_{4,3}$. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $w'_5 - w'_3$.

As a result, we get the following branching vector:

$$(w'_5 + w'_4 + 3m_4 + 2m_1,\ w'_5 - w'_3). \tag{24}$$

Case c-5. None of the previous cases are applicable, and there exist vertices $v \in V_{\mathrm{f5}}$ and $t_1 \in N_U(v; V_{\mathrm{f4}})$ such that $N_U(v) \cap N_U(t_1) \neq \emptyset$. We distinguish two sub-cases, according to the cardinality of the intersection $N_U(v) \cap N_U(t_1)$,

c-5(I) $|N_U(v) \cap N_U(t_1)| = 1$, and

c-5(II) $|N_U(v) \cap N_U(t_1)| = 2$.

Case c-5(I). Without loss of generality, assume that $N_U(v) \cap N_U(t_1) = \{t_2\}$ (see **Fig. 7**): We branch on edge vt_1. Note that $N_U(t_1) \setminus \{v\} = \{t_5\}$.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and edges vt_2, vt_3, vt_4, t_1t_2, and t_1t_5 will be deleted from G' by the reduction rules. So, the weight of vertex v decreases by w'_5 and the weight of vertex t_1 decreases by w'_4. The vertex t_2 must be one of types f4, u4, f5, and a u5-vertex, and its weight decreases by at least $m_5 = \min\{w'_4, w_4, \Delta'_{5,3}, \Delta_{5,3}\}$. Each of the vertices t_3 and t_4 must be one of types f4, u4, f5, and a u5-vertex, and each of their weights decreases by at least $m_4 = \min\{\Delta'_{4,3}, \Delta_{4,3}, \Delta'_{5,4}, \Delta_{5,4}\}$. The vertex t_5 must be one of types f3, u3, f4, u4, f5, and a u5-vertex, and its weight decreases by at least $m_1 = \min\{w'_3, w_3, \Delta'_{4,3}, \Delta_{4,3}, \Delta'_{5,4}, \Delta_{5,4}\}$. Thus, the total weight decrease in the branch of **force**(vt_1) is at least

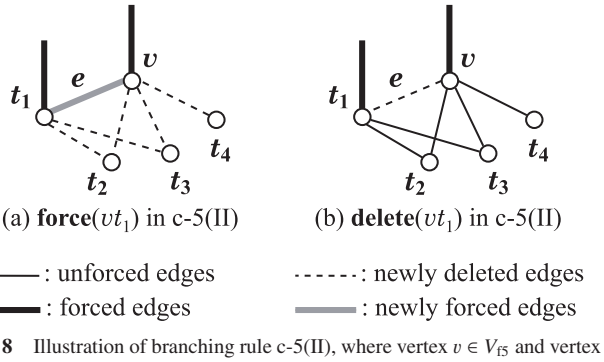

Fig. 8 Illustration of branching rule c-5(II), where vertex $v \in V_{\mathrm{f5}}$ and vertex $t_1 \in N_U(v; V_{\mathrm{f4}})$ such that $N_U(v) \cap N_U(t_1) = \{t_2, t_3\}$.

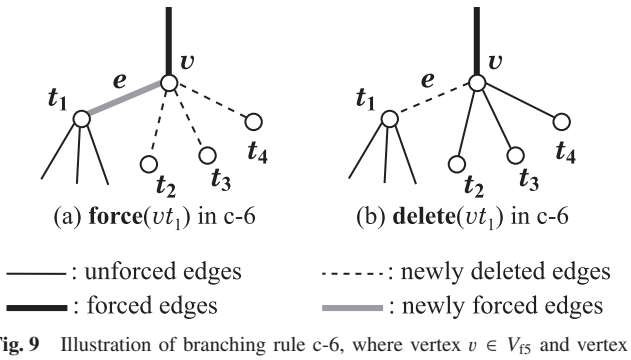

Fig. 9 Illustration of branching rule c-6, where vertex $v \in V_{\mathrm{f5}}$ and vertex $t_1 \in N_U(v; V_{\mathrm{u4}})$.

$w'_5 + w'_4 + m_5 + 2m_4 + m_1$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation. So, the weight of vertex v decreases by $\Delta'_{5,4}$, and the weight of vertex t_1 decreases by $\Delta'_{4,3}$. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $w'_5 - w'_3$.

As a result, we get the following branching vector:

$$(w'_5 + w'_4 + m_5 + 2m_4 + m_1,\ w'_5 - w'_3). \tag{25}$$

Case c-5(II). Without loss of generality, assume that $N_U(v) \cap N_U(t_1) = \{t_2, t_3\}$ (see **Fig. 8**): We branch on edge vt_1.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and edges vt_2, vt_3, vt_4, t_1t_2 and t_1t_3 will be deleted from G' by the reduction rules. So, the weight of vertex v decreases by w'_5 and the weight of vertex t_1 decreases by w'_4. Each of the vertices t_2 and t_3 must be one of types f4, u4, f5, and a u5-vertex, and each of their weights decreases by at least $m_5 = \min\{w'_4, w_4, \Delta'_{5,3}, \Delta_{5,3}\}$. The vertex t_4 must be one of types f4, u4, f5, and a u5-vertex, and its weight decreases by at least $m_4 = \min\{\Delta'_{4,3}, \Delta_{4,3}, \Delta'_{5,4}, \Delta_{5,4}\}$. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $w'_5 + w'_4 + 2m_5 + m_4$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation. So, the weight of vertex v decreases by $\Delta'_{5,4}$, and the weight of vertex t_1 decreases by $\Delta'_{4,3}$. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $w'_5 - w'_3$.

As a result, we get the following branching vector:

$$(w'_5 + w'_4 + 2m_5 + m_4,\ w'_5 - w'_3). \tag{26}$$

Case c-6. None of the previous cases are applicable, and there exist vertices $v \in V_{\mathrm{f5}}$ and $t_1 \in N_U(v; V_{\mathrm{u4}})$ (see **Fig. 9**): We branch on edge vt_1.

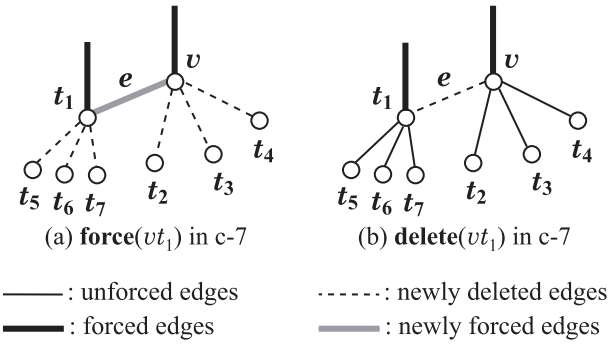

Fig. 10 Illustration of branching rule c-7, where vertex $v \in V_{\mathrm{f5}}$ and vertex $t_1 \in N_U(v; V_{\mathrm{f5}})$, such that $N_U(v) \cap N_U(t_1) = \emptyset$.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and edges vt_2, vt_3 and vt_4 will be deleted from G' by the reduction rules. So, the weight of vertex v decreases by w'_5 and the weight of vertex t_1 decreases by Δ_4. Each of the vertices t_2, t_3 and t_4 must be one of types u4, f5, and a u5-vertex, and each of their weights decreases by at least $m_6 = \min\{\Delta_{4,3}, \Delta'_{5,4}, \Delta_{5,4}\}$. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $w'_5 + w_4 - w'_4 + 3m_6$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation. So, the weight of vertex v decreases by $\Delta'_{5,4}$ and the weight of vertex t_1 decreases by $\Delta_{4,3}$. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $w'_5 - w'_4 + w_4 - w_3$.

As a result, we get the following branching vector:

$$(w'_5 + w_4 - w'_4 + 3m_6,\ w'_5 - w'_4 + w_4 - w_3). \tag{27}$$

Case c-7. None of the previous cases are applicable, and there exist vertices $v \in V_{\mathrm{f5}}$ and $t_1 \in N_U(v; V_{\mathrm{f5}})$ such that $N_U(v) \cap N_U(t_1) = \emptyset$ (see **Fig. 10**): We branch on edge vt_1. Note that $N_U(t_1) \setminus \{v\} = \{t_5, t_6, t_7\}$.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and edges vt_2, vt_3, vt_4, t_1t_5, t_1t_6 and t_1t_7 will be deleted from G' by the reduction rules. So, both weights of vertex v and vertex t_1 each decrease by w'_5. Each of the vertices t_2, t_3, t_4, t_5, t_6 and t_7 must be a type f5, or a u5-vertex, and each of their weights decreases by at least $m_7 = \min\{\Delta'_{5,4}, \Delta_{5,4}\}$. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $2w'_5 + 6m_7$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation. So, both weights of vertices v and t_1 each decrease by $\Delta'_{5,4}$. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $2w'_5 - 2w'_4$.

As a result, we get the following branching vector:

$$(2w'_5 + 6m_7,\ 2w'_5 - 2w'_4). \tag{28}$$

Case c-8. None of the previous cases are applicable, and there exist vertices $v \in V_{\mathrm{f5}}$ and $t_1 \in N_U(v; V_{\mathrm{f5}})$ such that $N_U(v) \cap N_U(t_1) \neq \emptyset$. We distinguish three sub-cases, according to the cardinality of the intersection $N_U(v) \cap N_U(t_1)$,

c-8(I) $|N_U(v) \cap N_U(t_1)| = 1$,
c-8(II) $|N_U(v) \cap N_U(t_1)| = 2$, and
c-8(III) $|N_U(v) \cap N_U(t_1)| = 3$.

Case c-8(I). Without loss of generality, assume that $N_U(v) \cap N_U(t_1) = \{t_2\}$ (see **Fig. 11**): We branch on edge vt_1. Note that $N_U(t_1) \setminus \{v\} = \{t_5, t_6\}$.

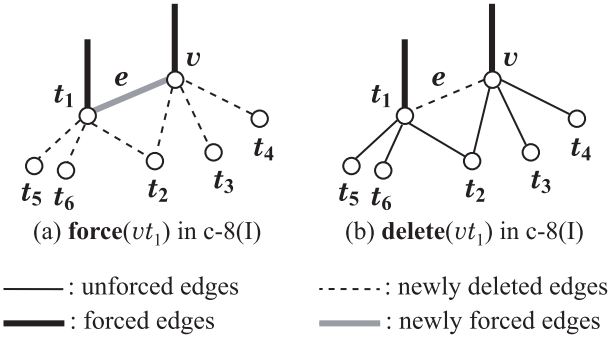

Fig. 11 Illustration of branching rule c-8(I), where vertex $v \in V_{\mathrm{f5}}$ and vertex $t_1 \in N_U(v; V_{\mathrm{f5}})$, such that $N_U(v) \cap N_U(t_1) = \{t_2\}$.

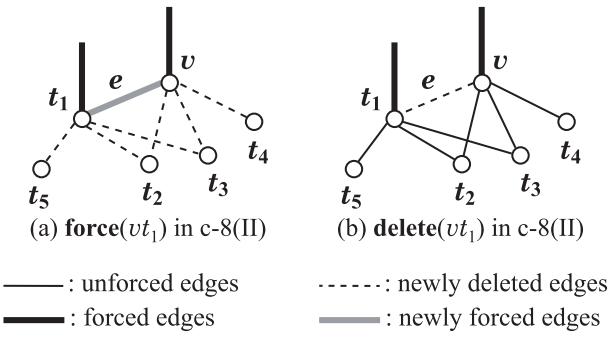

Fig. 12 Illustration of branching rule c-8(II), where vertex $v \in V_{\mathrm{f5}}$ and vertex $t_1 \in N_U(v; V_{\mathrm{f5}})$, such that $N_U(v) \cap N_U(t_1) = \{t_2, t_3\}$.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and edges vt_2, vt_3, vt_4, t_1t_2, t_1t_5 and t_1t_6 will be deleted from G' by the reduction rules. So, both weights of vertex v and vertex t_1 each decrease by w'_5. The vertex t_2 must be a type f5 or a u5-vertex, and its weight decreases by at least $m_8 = \min\{\Delta'_{5,3}, \Delta_{5,3}\}$. Each of the vertices t_3, t_4, t_5 and t_6 must be a type f5, or a u5-vertex, and each of their weights decreases by at least $m_7 = \min\{\Delta'_{5,4}, \Delta_{5,4}\}$. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $2w'_5 + 4m_7 + m_8$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation. So, both weights of vertices v and t_1 each decreases by $\Delta'_{5,4}$. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $2w'_5 - 2w'_4$.

As a result, we get the following branching vector:

$$(2w'_5 + 4m_7 + m_8,\ 2w'_5 - 2w'_4). \tag{29}$$

Case c-8(II). Without loss of generality, assume that $N_U(v) \cap N_U(t_1) = \{t_2, t_3\}$ (see **Fig. 12**): We branch on edge vt_1. Note that $N_U(t_1) \setminus \{v\} = \{t_5\}$.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and edges vt_2, vt_3, vt_4, t_1t_2, t_1t_3 and t_1t_5 will be deleted from G' by the reduction rules. So, both weights of vertex v and vertex t_1 each decrease by w'_5. Each of the vertices t_2 and t_3 must be a type f5, or a u5-vertex, and each of their weights decreases by at least $m_8 = \min\{\Delta'_{5,3}, \Delta_{5,3}\}$. Each of the vertices t_4 and t_5 must be a type f5, or a u5-vertex, and each of their weights decreases by at least $m_7 = \min\{\Delta'_{5,4}, \Delta_{5,4}\}$. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $2w'_5 + 2m_8 + 2m_7$.

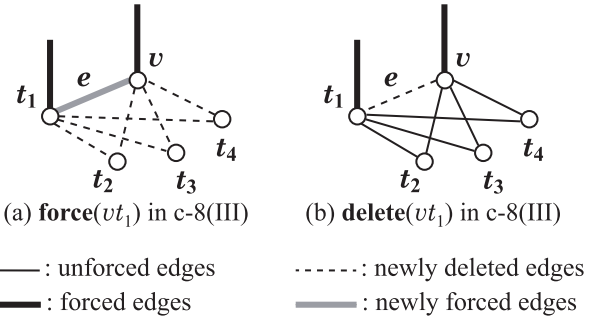

Fig. 13 Illustration of branching rule c-8(III), where vertex $v \in V_{\mathrm{f5}}$ and vertex $t_1 \in N_U(v; V_{\mathrm{f5}})$, such that $N_U(v) \cap N_U(t_1) = \{t_2, t_3, t_4\}$.

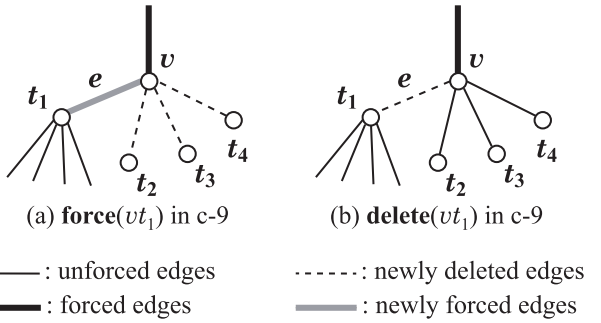

Fig. 14 Illustration of branching rule c-9, where vertex $v \in V_{\mathrm{f5}}$ and vertex $t_1 \in N_U(v; V_{\mathrm{u5}})$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation. So, both weights of vertex v and vertex t_1 each decreases by $\Delta'_{5,4}$. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $2w'_5 - 2w'_4$.

As a result, we get the following branching vector:

$$(2w'_5 + 2m_8 + 2m_7,\ 2w'_5 - 2w'_4). \tag{30}$$

Case c-8(III). We have that $N_U(v) \cap N_U(t_1) = \{t_2, t_3, t_4\}$ (see **Fig. 13**): We branch on edge vt_1.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and edges vt_2, vt_3, vt_4, t_1t_2, t_1t_3 and t_1t_4 will be deleted from G' by the reduction rules. So, both weights of vertex v and vertex t_1 each decrease by w'_5. Each of the vertices t_2, t_3, and t_4 must be a type f5, or a u5-vertex, and each of their weights decreases by at least $m_8 = \min\{\Delta'_{5,3}, \Delta_{5,3}\}$. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $2w'_5 + 3m_8$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation. So, both weights of vertex v and vertex t_1 each decrease by $\Delta'_{5,4}$. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $2w'_5 - 2w'_4$.

As a result, we get the following branching vector:

$$(2w'_5 + 3m_8,\ 2w'_5 - 2w'_4). \tag{31}$$

Case c-9. None of the previous cases are applicable, and there exist vertices $v \in V_{\mathrm{f5}}$ and $t_1 \in N_U(v; V_{\mathrm{u5}})$ (see **Fig. 14**): We branch on edge vt_1.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and edges vt_2, vt_3 and vt_4 will be deleted from G' by the reduction rules. So, the weight of vertex v decreases by w'_5, and the weight of vertex t_1 decreases by Δ_5. Each of the vertices t_2, t_3 and t_4 must be a type u5-vertex, and each of their weights decrease by $\Delta_{5,4}$. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $4w_5 - 3w_4$.

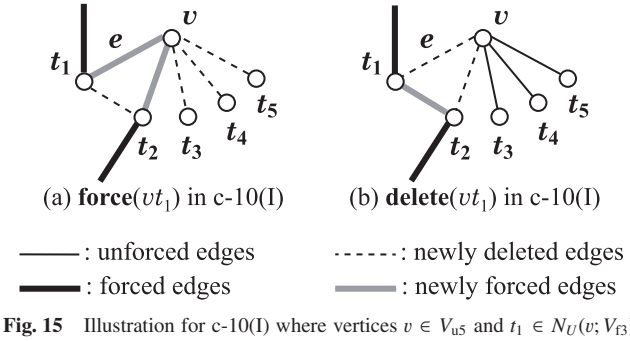

Fig. 15 Illustration for c-10(I) where vertices $v \in V_{\mathrm{u5}}$ and $t_1 \in N_U(v; V_{\mathrm{f3}})$, such that $N_U(v) \cap N_U(t_1) = \{t_2\}$, t_2 is an f3-vertex.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation. Thus, the weight of vertex v decreases by $\Delta'_{5,4}$, and the weight of vertex t_1 decreases by $\Delta_{5,4}$. The total weight decrease in the branch of **delete**(vt_1) is at least $w_5 + w'_5 - w_4 - w'_4$.

Then, we get the following branching vector:

$$(4w_5 - 3w_4,\ w_5 + w'_5 - w_4 - w'_4). \tag{32}$$

3.5 Branching on Edges Around u5-vertices (Branching Rules c-10 to c-15)

If none of the first nine conditions can be executed, this means that the instance of forced TSP has no f5-vertices. But this does not mean that the maximum degree of the reduced instance at this point is at most 4, since there might still be u5-vertices. This section derives branching vectors for branchings on an optimal edge $e = vt_1$ incident to a u5-vertex v, distinguishing the conditions of six cases, c-10 to c-15.

When branching on an edge vt_1 incident to a u5-vertex v, let a and b be lower bound on the decrease in measure due to the **force**(e) and **delete**(e) operations, respectively. In the branch of **force**(vt_1), the vertex v will become an f5-vertex. Therefore, at this point, some of the branching rules of cases c-1 to c-9 become immediately applicable. Let (A, B) be the branching vectors of the branching rules that become immediately applicable. Then we regard the overall branching vectors to be $(a + A, a + B, b)$ [7].

Case c-10. There are no more f5-vertices, and there exist vertices $v \in V_{\mathrm{u5}}$ and $t_1 \in N_U(v; V_{\mathrm{f3}})$ such that $N_U(v) \cap N_U(t_1) \neq \emptyset$. Note that it holds $|N_U(v) \cap N_U(t_1)| = 1$. Without loss of generality, assume that $N_U(v) \cap N_U(t_1) = \{t_2\}$. We distinguish five sub-cases, according to the type of vertex t_2, the unique vertex in the intersection $N_U(v) \cap N_U(t_1)$,

- c-10(I) t_2 is an f3-vertex,
- c-10(II) t_2 is a u3-vertex,
- c-10(III) t_2 is an f4-vertex,
- c-10(IV) t_2 is a u4-vertex, and
- c-10(V) t_2 is a u5-vertex.

Case c-10(I). Vertex t_2 is an f3-vertex (see **Fig. 15**): We branch on edge vt_1.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, edges vt_3, vt_4, vt_5 and t_1t_2 will be deleted from G' by the reduction rules, and the edge vt_2 will be added to F' by the reduction rules. Hence, the weight of vertex v decreases by w_5, and the weight of each of the vertices t_1 and t_2

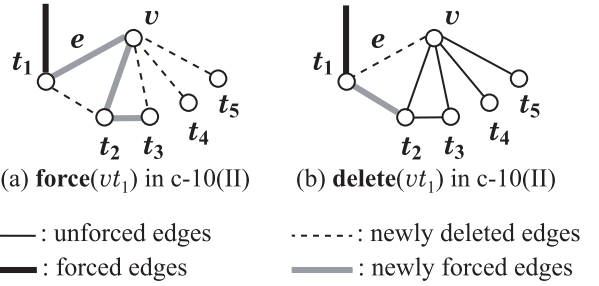

Fig. 16 Illustration for c-10(II) where vertices $v \in V_{u5}$ and $t_1 \in N_U(v; V_{f3})$, such that $N_U(v) \cap N_U(t_1) = \{t_2\}$, t_2 is a u3-vertex, and the unique neighbor of t_2 not in $\{t_1, v\}$ is in $N_U(v)$.

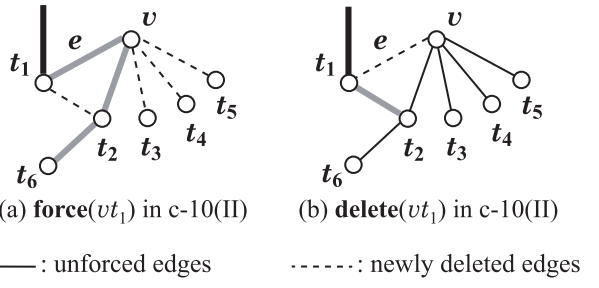

Fig. 17 Illustration for c-10(II) where vertices $v \in V_{u5}$ and $t_1 \in N_U(v; V_{f3})$, such that $N_U(v) \cap N_U(t_1) = \{t_2\}$, t_2 is a u3-vertex, and the unique neighbor of t_2 not in $\{t_1, v\}$ is not in $N_U(v)$.

decrease by w'_3. Each of the vertices t_3, t_4 and t_5 must be one of types f3, u3, f4, u4, and a u5-vertex, and each of their weights decreases by at least $m_{10} = \min\{w'_3, w_3, \Delta'_{4,3}, \Delta_{4,3}, \Delta_{5,4}\}$. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $(w_5 + 2w'_3 + 3m_{10})$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation, the edge vt_2 will be deleted from G' by the reduction rules, and the edge t_1t_2 will be added to F' by the reduction rules. Hence, the weight of vertex v decreases by $\Delta_{5,3}$, and both weights of vertices t_1 and t_2 each decrease by w'_3. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $w_5 - w_3 + 2w'_3$.

As a result, we get the following branching vector:

$$(w_5 + 2w'_3 + 3m_{10},\ w_5 - w_3 + 2w'_3). \tag{33}$$

Case c-10(II). Vertex t_2 is a u3-vertex (see **Figs. 16** and **17**): We branch on edge vt_1. Note that $N_U(t_2) \setminus \{v, t_1\} = \{t_6\}$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation, and the edge t_1t_2 will be added to F' by the reduction rules. Hence, the weight of vertex v decreases by $\Delta_{5,4}$, and the weight of vertex t_1 decreases by w'_3, and the weight of vertex t_2 decreases by Δ_3. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $w_5 - w_4 + w'_3 + w_3 - w'_3$.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, the edge vt_2 will be added to F' by the reduction rules, and edges vt_3, vt_4, vt_5 and t_1t_2 will be deleted from G' by the reduction rules.

For vertex t_6 it holds either a) $t_6 \in N_U(v)$ (see Fig. 16), or b) $t_6 \notin N_U(v)$ (see Fig. 17).

If the vertex $t_6 \in N_U(v) \cap N_U(t_2)$, then without loss of generality we assume that t_6 is t_3 (see Fig. 16). Then, the edge t_2t_3 will be added to F' by the reduction rules. Hence, the weight of

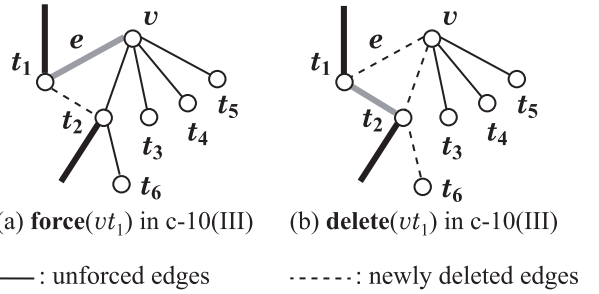

Fig. 18 Illustration for c-10(III) where vertices $v \in V_{u5}$ and $t_1 \in N_U(v; V_{f3})$, such that $N_U(v) \cap N_U(t_1) = \{t_2\}$, t_2 is an f4-vertex, and $N_U(v) \cap N_U(t_2) \setminus \{t_1\} = \emptyset$.

vertex v decreases by w_5, the weight of vertex t_1 decreases by w'_3, and the weight of vertex t_2 decreases by w_3. The vertex t_3 must be one of types f3, u3, f4, u4, and a u5-vertex, and its weight decreases by at least $m_{11} = \min\{w'_3, w_3, w'_4, w_4 - w'_3, w_5 - w'_4\}$. Each of the vertices t_4 and t_5 must be one of types f3, u3, f4, u4, and a u5-vertex, and each of their weights decreases by at least $m_{10} = \min\{w'_3, w_3, \Delta'_{4,3}, \Delta_{4,3}, \Delta_{5,4}\}$. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $w_5 + w'_3 + w_3 + m_{11} + 2m_{10}$.

As a result, we get the following branching vector:

$$(w_5 + w'_3 + w_3 + m_{11} + 2m_{10},\ w_5 - w_4 + w_3). \tag{34}$$

Next, if the vertex $t_6 \notin N_U(v) \cap N_U(t_2)$ (see Fig. 17), then the edge t_2t_6 will be added to F' by the reduction rules. Hence, the weight of vertex v decreases by w_5, the weight of vertex t_1 decreases by w'_3, and the weight of vertex t_2 decreases by w_3. The vertex t_6 must be one of types f3, u3, f4, u4, and a u5-vertex, and its weight decreases by at least $m_9 = \min\{w'_3, \Delta_3, w'_4, \Delta_4, \Delta_5\}$. Each of the vertices t_3, t_4 and t_5 must be one of types f3, u3, f4, u4, and a u5-vertex, and each of their weights decreases by at least $m_{10} = \min\{w'_3, w_3, \Delta'_{4,3}, \Delta_{4,3}, \Delta_{5,4}\}$. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $w_5 + w_3 + w'_3 + m_9 + 3m_{10}$.

As a result, we get the following branching vector:

$$(w_5 + w_3 + w'_3 + m_9 + 3m_{10},\ w_5 - w_4 + w_3). \tag{35}$$

Case c-10(III). Vertex t_2 is an f4-vertex (see **Fig. 18**): We branch on edge vt_1. Note that $N_U(t_2) \setminus \{v, t_1\} = \{t_6\}$.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and the edge t_1t_2 will be deleted from G' by the reduction rules. Hence, the weight of vertex v decreases by Δ_5, the weight of vertex t_1 decreases by w'_3, and the weight of vertex t_2 decreases by $\Delta'_{4,3}$. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $w_5 - w'_5 + w'_3 + w'_4 - w'_3$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation, the edge t_1t_2 will be added to F' by the reduction rules, and edges vt_2 and t_2t_6 will be deleted from G' by the reduction rules. Hence, the weight of vertex v decreases by $\Delta_{5,3}$, the weight of vertex t_1 decreases by w'_3, and the weight of vertex t_2 decreases by w'_4. The vertex t_6 must be one of types f3, u3, f4, u4, and a u5-vertex, and its weight decreases by at least $m_{10} = \min\{w'_3, w_3, \Delta'_{4,3}, \Delta_{4,3}, \Delta_{5,4}\}$. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $w_5 - w_3 + w'_3 + w'_4 + m_{10}$.

As a result, we get the following branching vector:

$$(w_5 - w'_5 + w'_4,\ w_5 - w_3 + w'_3 + w'_4 + m_{10}).$$

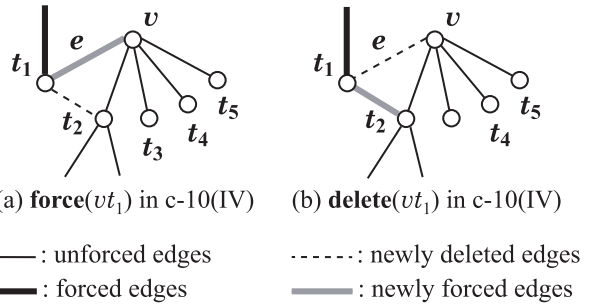

Fig. 19 Illustration for c-10(IV) where vertices $v \in V_{u5}$ and $t_1 \in N_U(v; V_{f3})$, such that $N_U(v) \cap N_U(t_1) = \{t_2\}$, t_2 is a u4-vertex.

In the branch of **force**(vt_1), the vertex v will become an f5-vertex, and at this point, one of the branching rules of cases c-1 and c-2 becomes immediately applicable, because an optimal edge for the next branching operation is at least as good as the edge vt_2. Therefore, for the case where the vertex t_3 is either an f3 or a u3-vertex, for each of the 19 branching vectors (A, B) of cases c-1 and c-2, we get the following branching vector:

$$(w_5 - w'_5 + w'_4 + A,\ w_5 - w'_5 + w'_4 + B,\ w_5 - w_3 + w'_3 + w'_4 + m_{10}). \tag{36}$$

Case c-10(IV). Vertex t_2 is a u4-vertex (see **Fig. 19**): We branch on edge vt_1.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and the edge t_1t_2 will be deleted from G' by the reduction rules. Hence, the weight of vertex v decreases by Δ_5, the weight of vertex t_1 decreases by w'_3, and the weight of vertex t_2 decreases by $\Delta_{4,3}$. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $w_5 - w'_5 + w'_3 + w_4 - w_3$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation, and the edge t_1t_2 will be added to F' by the reduction rules. Hence, the weight of vertex v decreases by $\Delta_{5,4}$, the weight of vertex t_1 decreases by w'_3, and the weight of vertex t_2 decreases by Δ_4. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $w_5 - w_4 + w'_3 + w_4 - w'_4$.

As a result, we get the following branching vector:

$$(w_5 - w'_5 + w'_3 + w_4 - w_3,\ w_5 + w'_3 - w'_4).$$

In the branch of **force**(vt_1), the vertex v will become an f5-vertex, and at this point, one of the branching rules of cases c-1, c-2 and c-3 becomes immediately applicable, because an optimal edge for the next branching operation is at least as good as the edge vt_2. Therefore, for each of the 20 branching vectors (A, B) of cases c-1 to c-3, we get the following branching vector:

$$(w_5 - w'_5 + w'_3 + w_4 - w_3 + A,\ w_5 - w'_5 + w'_3 + w_4 - w_3 + B,\ w_5 + w'_3 - w'_4). \tag{37}$$

Case c-10(V). Vertex t_2 is a u5-vertex (see **Fig. 20**): We branch on edge vt_1.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and the edge t_1t_2 will be deleted from G' by the reduction rules. Hence, the weight of vertex v decreases by Δ_5, the weight of vertex t_1 decreases by w'_3, and the weight of vertex t_2 decreases by $\Delta_{5,4}$. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $w_5 - w'_5 + w'_3 + w_5 - w_4$.

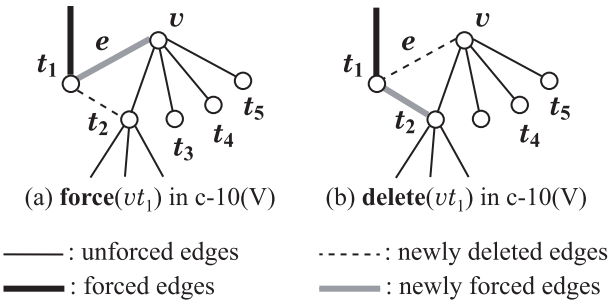

Fig. 20 Illustration for c-10(V) where vertices $v \in V_{u5}$ and $t_1 \in N_U(v; V_{f3})$, such that $N_U(v) \cap N_U(t_1) = \{t_2\}$, t_2 is a u5-vertex.

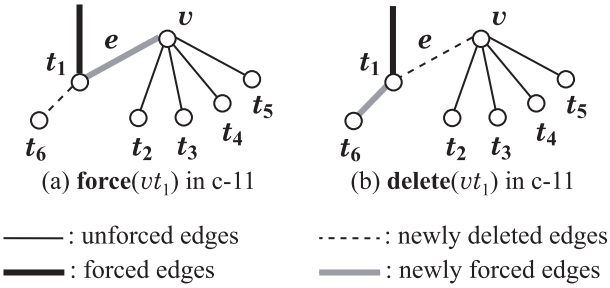

Fig. 21 Illustration for c-11 where vertices $v \in V_{u5}$ and $t_1 \in N_U(v; V_{f3})$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation, and the edge t_1t_2 will be added to F' by the reduction rules. Hence, the weight of vertex v decreases by $\Delta_{5,4}$, the weight of vertex t_1 decreases by w'_3, and the weight of vertex t_2 decreases by Δ_5. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $w_5 - w_4 + w'_3 + w_5 - w'_5$.

As a result, we get the following branching vector:

$$(2w_5 - w'_5 + w'_3 - w_4,\ 2w_5 - w_4 + w'_3 - w'_5).$$

In the branch of **force**(vt_1), the vertex v will become an f5-vertex, and at this point, one of the branching rules of cases c-1 to c-6 becomes immediately applicable, because an optimal edge for the next branching operation is at least as good as the edge vt_2. Therefore, for each of the 24 branching vectors (A, B) of cases c-1 to c-6, we get the following branching vector:

$$(2w_5 - w'_5 + w'_3 - w_4 + A,\ 2w_5 - w'_5 + w'_3 - w_4 + B,\ 2w_5 - w_4 + w'_3 - w'_5). \tag{38}$$

Case c-11. None of the previous cases are applicable, and there exist vertices $v \in V_{u5}$ and $t_1 \in N_U(v; V_{f3})$ such that $N_U(v) \cap N_U(t_1) = \emptyset$ (see **Fig. 21**): We branch on edge vt_1. Note that $N_U(t_1) \setminus \{v\} = \{t_6\}$.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and the edge t_1t_6 will be deleted from G' by the reduction rules. So, the weight of vertex v decreases by Δ_5, and the weight of vertex t_1 decreases by w'_3.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation, and the edge t_1t_6 will be added to F' by the reduction rules. The weight of vertex v decreases by $\Delta_{5,4}$, and the weight of vertex t_1 decreases by w'_3.

There are two cases for the vertex t_6; a) vertex t_6 is of type f3, and b) otherwise. We analyze the branches **force**(vt_1) and **delete**(vt_1) for these two cases separately.

First, we analyze the case where the vertex t_6 is an f3-vertex

(see Fig. 2). Recall that in such a case we denote by x the unique neighbor of t_6 different from t_1. In the branch of **force**(vt_1), the edge xt_6 will be added to F' by the reduction rules. Hence, the weight of vertex t_6 decreases by w'_3. If the vertex x is an f3-vertex (resp., u3, f4, u4, and a u5-vertex), then the weight decrease α_5 of vertex x will be w'_3 (resp., Δ_3, w'_4, Δ_4, and Δ_5). Thus, the total weight decrease for this case in the branch of **force**(vt_1) is at least $w_5 - w'_5 + w'_3 + w'_3 + \alpha_5$.

In the branch of **delete**(vt_1), the edge xt_6 will be deleted from G' by the reduction rules. Hence, the weight of vertex t_6 decreases by w'_3. If the vertex x is an f3-vertex (resp., u3, f4, u4, and a u5-vertex), then the weight decrease β_5 of vertex x will be w'_3 (resp., w_3, $\Delta'_{4,3}$, $\Delta_{4,3}$, and $\Delta_{5,4}$). Thus, the total weight decrease for the case where t_6 is an f3-vertex in the branch of **delete**(vt_1) is at least $w_5 - w_4 + w'_3 + w'_3 + \beta_5$.

As a result, for the ordered pair (α_5, β_5) taking values in $\{(w'_3, w'_3), (\Delta_3, w_3), (w'_4, \Delta'_{4,3}), (\Delta_4, \Delta_{4,3}), (\Delta_5, \Delta_{5,4})\}$, we get the following five branching vectors:

$$(w_5 - w'_5 + 2w'_3 + \alpha_5,\ w_5 - w_4 + 2w'_3 + \beta_5).$$

In the branch of **force**(vt_1), the vertex v will become an f5-vertex, and at this point, one of the branching rules of cases c-1 to c-9 becomes immediately applicable. Therefore, for the case where the vertex t_6 is an f3-vertex, for each of the 29 branching vectors (A, B) of cases c-1 to c-9, we get the following branching vector:

$$(w_5 - w'_5 + 2w'_3 + \alpha_5 + A,\ w_5 - w'_5 + 2w'_3 + \alpha_5 + B,\ w_5 - w_4 + 2w'_3 + \beta_5). \tag{39}$$

Next, we examine the case where the vertex t_6 is not an f3-vertex. In the branch of **force**(vt_1), if the vertex t_6 is a u3-vertex (resp., f4, u4, and a u5-vertex), then the weight decrease α_6 of vertex t_6 will be w_3 (resp., $\Delta'_{4,3}$, $\Delta_{4,3}$, and $\Delta_{5,4}$). Thus, the total weight decrease for this case in the branch of **force**(vt_1) is at least $w_5 - w'_5 + w'_3 + \alpha_6$.

In the branch of **delete**(vt_1), if the vertex t_6 is a u3-vertex (resp., f4, u4, and a u5-vertex), then the weight decrease β_6 of vertex t_6 will be Δ_3 (resp., w'_4, Δ_4, and Δ_5). Thus, the total weight decrease for the case where t_6 is not an f3-vertex in the branch of **delete**(vt_1) is at least $w_5 - w_4 + w'_3 + \beta_6$.

As a result, for the ordered pair (α_6, β_6) taking values in $\{(w_3, \Delta_3), (\Delta'_{4,3}, w'_4), (\Delta_{4,3}, \Delta_4), (\Delta_{5,4}, \Delta_5)\}$, we get the following four branching vectors:

$$(w_5 - w'_5 + w'_3 + \alpha_6,\ w_5 - w_4 + w'_3 + \beta_6).$$

In the branch of **force**(vt_1), for each of the 29 branching vectors (A, B) of cases c-1 to c-9, we get the following branching vector:

$$(w_5 - w'_5 + w'_3 + \alpha_6 + A,\ w_5 - w'_5 + w'_3 + \alpha_6 + B,\ w_5 - w_4 + w'_3 + \beta_6). \tag{40}$$

Case c-12. None of the previous cases are applicable, and there exist vertices $v \in V_{u5}$ and $t \in N_U(v; V_{u3})$ (see **Fig. 22**): We branch on edge vt_1. Note that $N_U(t_1) \setminus \{v\} = \{t_6, t_7\}$.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation. So, the weight of vertex v decreases by

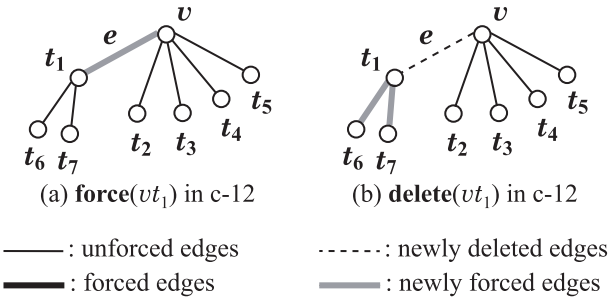

Fig. 22 Illustration of branching rule c-12, where vertex $v \in V_{u5}$ and vertex $t \in N_U(v; V_{u3})$.

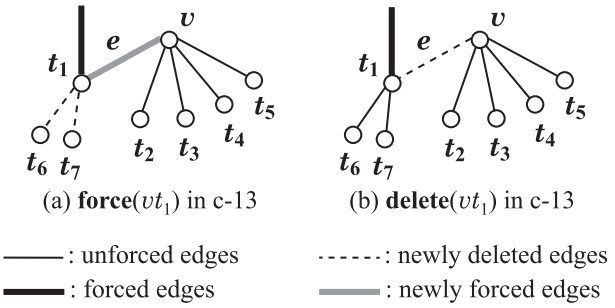

Fig. 23 Illustration for c-13 where vertices $v \in V_{u5}$ and $t_1 \in N_U(v; V_{f4})$.

Δ_5, and the weight of vertex t_1 decreases by Δ_3. The total weight decrease in the branch of **force**(vt_1) is at least $w_5 - w'_5 + w_3 - w'_3$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation, and edges t_1t_6 and t_1t_7 will be added to F' by the reduction rules. So, the weight of vertex v decreases by $\Delta_{5,4}$, and the weight of vertex t_1 decreases by w_3. Each of the vertices t_6 and t_7 must be one of types f3, u3, f4, u4, and a u5-vertex, and each of their weights decreases by at least $m_9 = \min\{w'_3, \Delta_3, w'_4, \Delta_4, \Delta_5\}$. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $w_5 - w_4 + w_3 + 2m_9$.

As a result, we get the following branching vector:

$$(w_5 - w'_5 + w_3 - w'_3,\ w_5 - w_4 + w_3 + 2m_9).$$

In the branch of **force**(vt_1), the vertex v will become an f5-vertex, and at this point, one of the branching rules of cases c-3 to c-6 and c-9 becomes immediately applicable, since there are no more f3 vertices and no other f5 vertices, thus cases c-1, c-2, c-7 and c-8 cannot occur following this branching operation. Therefore, for each of the six branching vectors (A, B) of cases c-3 to c-6 and case c-9, we get the following branching vector:

$$(w_5 - w'_5 + w_3 - w'_3 + A,\ w_5 - w'_5 + w_3 - w'_3 + B,\ w_5 - w_4 + w_3 + 2m_9). \tag{41}$$

Case c-13. None of the previous cases are applicable, and there exist vertices $v \in V_{u5}$ and $t_1 \in N_U(v; V_{f4})$ (see **Fig. 23**): We branch on edge vt_1. Note that $N_U(t_1) \setminus \{v\} = \{t_6, t_7\}$.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation, and edges t_1t_6 and t_1t_7 will be deleted from G' by the reduction rules. So, the weight of vertex v decreases by Δ_5, and the weight of vertex t_1 decreases by w'_4. Each of the vertices t_6 and t_7 must be one of types f3, u3, f4, u4, and a u5-vertex, and each of their weights decreases by at least $m_{10} = \min\{w'_3, w_3, \Delta'_{4,3}, \Delta_{4,3}, \Delta_{5,4}\}$. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $w_5 - w'_5 + w'_4 + 2m_{10}$.

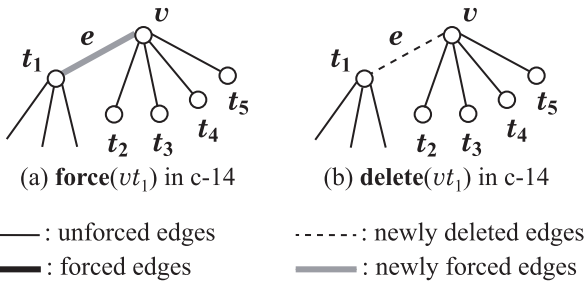

Fig. 24 Illustration of branching rule c-14, where vertex $v \in V_{u5}$ and vertex $t_1 \in N_U(v; V_{u4})$.

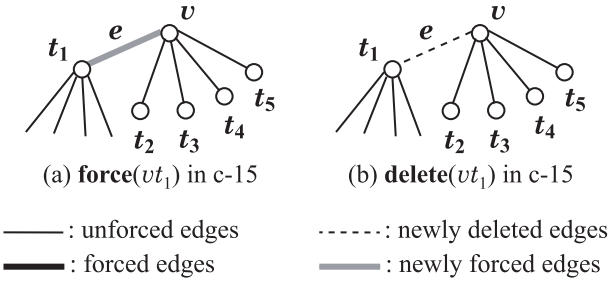

Fig. 25 Illustration of branching rule c-15, where vertex $v \in V_{u5}$ and vertex $t_1 \in N_U(v; V_{u5})$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation. So, the weight of vertex v decreases by $\Delta_{5,4}$, and the weight of vertex t_1 decreases by $\Delta'_{4,3}$. The total weight decrease in the branch of **delete**(vt_1) is at least $w_5 - w_4 + w'_4 - w'_3$.

As a result, we get the following branching vector:

$$(w_5 - w'_5 + w'_4 + 2m_{10},\ w_5 - w_4 + w'_4 - w'_3).$$

In the branch of **force**(vt_1), the vertex v will become an f5-vertex, and at this point, one of the branching rules of cases c-4 to c-9 becomes immediately applicable. Therefore, for each of the nine branching vectors (A, B) of cases c-4 to c-9, we get the following branching vector:

$$(w_5{-}w'_5{+}w'_4{+}2m_{10}{+}A,\ w_5{-}w'_5{+}w'_4{+}2m_{10}{+}B,\ w_5{-}w_4{+}w'_4{-}w'_3). \tag{42}$$

Case c-14. None of the previous cases are applicable, and there exist vertices $v \in V_{u5}$ and $t_1 \in N_U(v; V_{u4})$ (see **Fig. 24**): We branch on edge vt_1.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation. So, the weight of vertex v decreases by Δ_5, and the weight of vertex t_1 decreases by Δ_4. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $w_5 - w'_5 + w_4 - w'_4$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation. So, the weight of vertex v decreases by $\Delta_{5,4}$, and the weight of vertex t_1 decreases by $\Delta_{4,3}$. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $w_5 - w_3$.

Then, we get the following branching vector:

$$(w_5 - w'_5 + w_4 - w'_4,\ w_5 - w_3).$$

In the branch of **force**(vt_1), the vertex v will become an f5-vertex, and at this point, one of the branching rules of cases c-6 and c-9 becomes immediately applicable, since there are only u4 and u5 vertices that remain, and thus only cases c-6 and c-9 can occur following to this branching operation. Therefore, for each of the two branching vectors (A, B) of cases c-6 and c-9, we get the following branching vector:

$$(w_5 - w'_5 + w_4 - w'_4 + A,\ w_5 - w'_5 + w_4 - w'_4 + B,\ w_5 - w_3). \tag{43}$$

Case c-15. None of the previous cases are applicable, and there exist vertices $v \in V_{u5}$ and $t_1 \in N_U(v; V_{u5})$ (see **Fig. 25**): We branch on edge vt_1.

In the branch of **force**(vt_1), the edge vt_1 will be added to F' by the branching operation. So, both weights of vertex v and vertex t_1 each decrease by Δ_5. Thus, the total weight decrease in the branch of **force**(vt_1) is at least $2w_5 - 2w'_5$.

In the branch of **delete**(vt_1), the edge vt_1 will be deleted from G' by the branching operation. So, both weights of vertex v and vertex t_1 each decrease by $\Delta_{5,4}$. Thus, the total weight decrease in the branch of **delete**(vt_1) is at least $2w_5 - 2w_4$.

Then, we get the following branching vector:

$$(2w_5 - 2w'_5,\ 2w_5 - 2w_4).$$

In the branch of **force**(vt_1), the vertex v will become an f5-vertex, and at this point, the branching rule of case c-9 becomes immediately applicable, since there are only u5 vertices that remain, and thus only case c-9 can occur following to this branching operation. Therefore, for the branching vector (A, B) of case c-9, we get the following branching vector:

$$(2w_5 - 2w'_5 + A,\ 2w_5 - 2w'_5 + B,\ 2w_5 - 2w_4). \tag{44}$$

3.6 Switching to TSP in Degree-4 Graphs

If none of the above 15 cases can be executed, then this means that the graph has no more degree-5 vertices and the maximum degree of the graph at this point is 4 or less. In that case, we can switch and use a fast algorithm for TSP in degree-4 graphs, called tsp4(G, F) to solve the remaining instance $I = (G, F)$. Since at the point when tsp4(G, F) is called the instance $I = (G, F)$ is a reduced instance, it has no vertices of degree 1 or less, nor vertices with three or more incident forced edges. Moreover, without loss of generality we can assume that there is also no vertex $v \in V(G)$ of degree 2, as its weight $\omega(v)$ is set to be 0 in Eq. (18), and together with the two forced edges incident with v can be replaced by a single forced edge [15], [16]. Then, let n'_3, n_3, n'_4 and n_4 be the numbers of f3, u3, f4 and u4 vertices in I, respectively, where $n'_3 + n_3 + n'_4 + n_4 = |V(G)|$, and by Eq. (1) the measure $\mu(I)$ of instance I is $\mu(I) = w'_3 n'_3 + w_3 n_3 + w'_4 n'_4 + w_4 n_4$.

At this step, we switch to use the $O^*(1.69193^n)$-time algorithm for the TSP in degree-4 graphs by Xiao and Nagamochi [16]. Let $\hat{\mu}(I)$ denote the measure of the TSP in the degree-4 instance I, where $\hat{\mu}(I) = \hat{w}'_3 n'_3 + \hat{w}_3 n_3 + \hat{w}'_4 n'_4 + \hat{w}_4 n_4$, and the weight $\hat{w}'_3$ for an f3-vertex is 0.21968, the weight $\hat{w}_3$ for an u3-vertex is 0.45540, the weight $\hat{w}'_4$ for an f4-vertex is 0.59804 and the weight $\hat{w}_4$ for an u4-vertex is 1 [16]. Hence, the running time bound of the TSP in the degree-4 instance I with measure $\hat{\mu}(I)$ is given by $1.69193^{\hat{\mu}(I)}$. Since the bound on the running time of tsp5(G, F) is expressed

by $\tau^{\mu(I)}$, we get a constraint on the value of τ:

$$1.69193^{\hat{\mu}(I)} \leq \tau^{\mu(I)}$$

for any partition $\{n'_3, n_3, n'_4, n_4\}$ of $|V(G)|$. In other words,

$$1.69193^{0.21968} \leq \tau^{w'_3}, \quad (45)$$

$$1.69193^{0.45540} \leq \tau^{w_3}, \quad (46)$$

$$1.69193^{0.59804} \leq \tau^{w'_4}, \text{ and} \quad (47)$$

$$1.69193^{1} \leq \tau^{w_4}. \quad (48)$$

Equations (45) to (48) will become constraints in the quasiconvex program.

3.7 Overall Analysis

The branching factors of each of the branching vectors from Eqs. (19) to (44) and the switching constraints of Eqs. (45) to (48) do not exceed 2.349978. The tight constraints in the quasiconvex program are in conditions c-11, c-12, c-14, c-15 and the switching constraints. This completes the proof of Lemma 2, and therefore, Theorem 1.

4. Conclusion

In this paper, we have presented an exact algorithm for TSP in degree-5 graphs. Our algorithm is a simple branching algorithm, following the branch-and-reduce paradigm, and it can be implemented to operate in space which is polynomial of the size of an input instance. For the analysis of the running time of the proposed algorithm, we used the measure and conquer method, and have obtained an upper bound of $O^*(2.3500^n)$, where n is the number of vertices in a given instance. This result improves a previous time bound on the TSP in degree-5 graphs by Md Yunos et al. [13].

The greatest challenge in obtaining a non-trivial time-bound for the algorithm is to derive a proper case analysis, and the interdependence of the branching rules makes the problem ever more difficult with graphs of higher degree. In contrast, the above results offer a purely theoretically improved time-bound on the running time. With regards to a recent result by Akiba and Iwata [1], algorithms with theoretically improved running time can indeed be also superior in practice, and it would be of much interest to implement the proposed algorithms and evaluate their empirical performance. This branch of research would also open the question of devising bounding rules which might not have an impact on the theoretical running time bound, but might contribute meaningfully to the algorithm's performance in practice.

References

[1] Akiba, T. and Iwata, Y.: Branch-and-reduce Exponential/FPT Algorithms in Practice: A Case Study of Vertex Cover, *Proc. 17th Workshop on Algorithm Engineering and Experiments* (*ALENEX*), *SIAM*, pp.70–81 (2015).

[2] Bellman, R.E.: Combinatorial Processes and Dynamic Programming, *Combinatorial Analysis, Proc. Symposia in Applied Mathematics 10*, American Mathematical Society, pp.217–249 (1960).

[3] Bodlaender, H.L., Cygan, M., Kratsch, S. and Nederlof, J.: Deterministic Single Exponential Time Algorithms for Connectivity Problems Parameterized by Treewidth, *Information and Computation*, Vol.243, pp.86–111 (2015).

[4] Eppstein, D.: Quasiconvex Analysis of Backtracking Algorithms, *Proc. 15th Annual ACM-SIAM Symposium On Discrete Algorithms* (*SODA 2004*), pp.781–790, ACM Press (2004).

[5] Eppstein, D.: The Traveling Salesman Problem for Cubic Graphs, *Journal of Graph Algorithms and Application*, Vol.11, No.1, pp.61–81 (2007).

[6] Fomin, F.V., Grandoni, F. and Kratsch, D.: A Measure and Conquer Approach for the Analysis of Exact Algorithms, *J. ACM*, Vol.56, No.5, Article No.25 (2009).

[7] Fomin, F.V. and Kratsch, D.: *Exact Exponential Algorithms*, Springer (2010).

[8] Gebauer, H.: Finding and Enumerating Hamilton Cycles in 4-regular Graphs, *Theoretical Computer Science*, Vol.412, No.35, pp.4579–4591 (2011).

[9] Gurevich, Y. and Shelah, S.: Expected Computation Time for Hamiltonian Path Problem, *SIAM Journal of Computation*, Vol.16, No.3, pp.486–502 (1987).

[10] Held, M. and Karp, R.M.: A Dynamic Programming Approach to Sequencing Problems, *Journal of the Society for Industrial and Applied Mathematics*, Vol.10, No.1, pp.196–210 (1962).

[11] Iwama, K. and Nakashima, T.: An Improved Exact Algorithm for Cubic Graph TSP, *Computing and Combinatorics*, Lecture Notes in Computer Science, Vol.4598, pp.108–117 (2007).

[12] Liskiewicz, M. and Schuster, M.R.: A New Upper Bound for the Traveling Salesman Problem in Cubic Graphs, *Journal of Discrete Algorithms*, Vol.27, pp.1–20 (2014).

[13] Md Yunos, N., Shurbevski, A. and Nagamochi, H.: A Polynomial-space Exact Algorithm for TSP in Degree-5 Graphs, *The 12th International Symposium on Operations Research and Its Application* (*ISORA 2015*), pp.45–58 (2015).

[14] Rubin, F.: A Search Procedure for Hamilton Paths and Circuits, *J. ACM*, Vol.21, No.4, pp.576–580 (1974).

[15] Xiao, M. and Nagamochi, H.: An Exact Algorithm for TSP in Degree-3 Graphs via Circuit Procedure and Amortization on Connectivity Structure, *Algorithmica*, Vol.74, No.2, pp.713–741 (2016).

[16] Xiao, M. and Nagamochi, H.: An Improved Exact Algorithm for TSP in Graphs of Maximum Degree 4, *Theory of Computing Systems*, Vol.58, No.2, pp.241–272 (2016).

Norhazwani Md Yunos was born in 1985. She received her M.Sc. degree from the University of Technology, Malaysia in 2009, and a Ph.D. degree in Informatics from Kyoto University in 2017. Currently, she works at the Technical University of Malaysia Malacca, Malacca, Malaysia as a lecturer, resuming her position from before she left Malaysia to pursue her Ph.D. studies in Japan. Her research interests include Combinatorial Optimization, Routing and Scheduling, etc.

Aleksandar Shurbevski was born in 1986. He received his M.Sc. degree from Kyoto Institute of Technology in 2012, and a Ph.D. degree in Informatics from Kyoto University in 2014. As of 2014 he is an Assistant Professor at the Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, and his research interests include Combinatorial Optimization, Graph Algorithms, etc. He is a member of IEEE and JSOR.

Hiroshi Nagamochi was born in 1960. He received his D.E. degree from Kyoto University in 1988. He became a Professor in the Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University in 2004 after serving for Toyohashi University of Technology. His research interests include Discrete Optimization, Graph Algorithms, Scheduling, Packing Problems, Chemical Graphs, etc.