An Improved Randomized Approximation Algorithm for Max TSP

Zhi-Zhong Chen *   Lusheng Wang †

Abstract

We present an $O(n^3)$-time randomized approximation algorithm for the maximum traveling salesman problem whose expected approximation ratio is asymptotically $\frac{251}{331}$, where n is the number of vertices in the input (undirected) graph. This improves the previous best.

1 Introduction

The *maximum traveling salesman problem* (Max TSP) is to compute a maximum-weight Hamiltonian circuit (called a *tour*) in a given edge-weighted (undirected) graph. The problem is known to be Max-SNP-hard [1] and there have been a number of approximation algorithms known for it [3, 4, 7]. In 1984, Serdyukov [7] gave an $O(n^3)$-time approximation algorithm for Max TSP that achieves an approximation ratio of $\frac{3}{4}$. Serdyukov's algorithm is very simple and elegant, and it tempts one to ask if a better approximation ratio can be achieved for Max TSP by a polynomial-time approximation algorithm. Along this line, Hassin and Rubinstein [4] showed that with the help of randomization, better approximation ratio for Max TSP can be achieved. More precisely, they gave an $O(n^3)$-time randomized approximation algorithm for Max TSP whose *expected* approximation ratio is asymptotically $\frac{25}{33}$. Their algorithm is basically a combination of Serdyukov's algorithm and an earlier algorithm of their own [3].

The asymptotic ratio $\frac{25}{33}$ achieved by Hassin and Rubinstein's algorithm is marginally better than the ratio $\frac{3}{4}$ achieved by Serdyukov's algorithm. However, Hassin and Rubinstein said in their paper [4]: "the better ratio at least demonstrates that the ratio of $\frac{3}{4}$ can be improved and further research along this line is encouraged". Moreover, it is widely recognized that improving approximation algorithms for TSP and its variants are not easy. In this paper, following and improving Hassin and Rubinstein's work, we give a new $O(n^3)$-time randomized approximation algorithm for Max TSP whose expected approximation ratio is asymptotically $\frac{251}{331}$. Hassin and Rubinstein [4] show that each approximation algorithm for Max TSP can be translated into an approximation algorithm for a problem called the *maximum latency TSP* which was first studied by Chalasani and Motwani [2]. Using their translation, our new algorithm can be trivially turned into a new randomized approximation algorithm for the maximum latency TSP whose expected approximation ratio improves the previous best.

Like all previous approximation algorithms for Max TSP, our new algorithm starts by computing a maximum-weight cycle cover $\mathcal{C}$ of the input graph G and then modify the cycles in $\mathcal{C}$ (somehow) to a tour of G without losing much weight. All the previous algorithms modify the cycles in $\mathcal{C}$ in an arbitrary order. In contrast, our algorithm modify the cycles in a carefully chosen order based

*Department of Mathematical Sciences, Tokyo Denki University, Hatoyama, Saitama 350-0394, Japan. Email: chen@r.dendai.ac.jp. Part of work done during a visit at City University of Hong Kong.

†Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong.

on suitably constructed auxiliary graphs. Moreover, the way of modifying a cycle heavily depends on how the previous cycles were modified. This is why our algorithm is complicated.

After giving some basic definitions in Section 2, we sketch Hassin and Rubinstein's algorithm in Section 3. In Section 4, we describe our ideas for improving their algorithm. Section 5 contains an outline of our new algorithm. Section 6 details how to modify 4-cycles. Sections 7 and 8 detail how to modify non-4-cycles. Section 9 contains an analysis of the improved approximation ratio and the running time.

2 Basic Definitions

Throughout this paper, a graph means a simple undirected graph (i.e., it has neither parallel edges nor self-loops), while a multigraph may have parallel edges but no self-loops.

Let G be a graph. We denote the vertex set of G by $V(G)$, and denote the edge set of G by $E(G)$. In order to avoid confusion, we sometimes call the elements of $V(G)$ the *nodes* of G (rather than the vertices of G). For a subset U of $V(G)$, $G - U$ denotes the graph obtained from G by removing the vertices in U (together with the edges incident to them). For a subset F of $E(G)$, $G - F$ denotes the graph obtained from G by removing the edges in F. The *degree* of a vertex v in G is the number of edges incident to v in G. Two edges of G are *adjacent* if they have a common endpoint.

A *cycle* in G is a connected subgraph of G in which each vertex is of degree 2. A *path* in G is either a single vertex of G or a connected subgraph of G in which exactly two vertices are of degree 1 and the others are of degree 2. The *length* of a cycle or path C is the number of edges in C. A cycle is called a *k-cycle* if its length is k. If the length of a cycle or path P is odd, then we say that P is *odd*; otherwise, we say that P is *even*. A *tour* (also called a *Hamiltonian cycle*) of G is a cycle C of G with $V(C) = V(G)$. A *cycle cover* of G is a subgraph H of G with $V(H) = V(G)$ in which each vertex is of degree 2. A *subtour* of G is a subgraph H of G in which each connected component is a path.

A *matching* of G is a (possibly empty) set of pairwise nonadjacent edges of G. A *perfect matching* of G is a matching M of G such that each vertex of G is incident to an edge in M. An *independent set* of G is a (nonempty) set U of vertices in G such that no two vertices in U are adjacent in G. The *distance* between two vertices u and v in G is the length of the shortest path between u and v in G.

For a random event A, $\Pr[A]$ denotes the probability that A occurs. For two random events A and B, $\Pr[A \mid B]$ denotes the (conditional) probability that A occurs given the known occurrence of event B. For a random variable X, $\mathcal{E}[X]$ denotes the expected value of X.

Throughout the rest of the paper, fix an instance (G, w) of Max TSP, where G is a complete (undirected) graph and w is a function mapping each edge e of G to a nonnegative real number $w(e)$. For a subset F of $E(G)$, $w(F)$ denotes $\sum_{e \in F} w(e)$. The *weight* of a subgraph H of G is $w(H) = w(E(H))$. Our goal is to compute a tour of large weight in G. For ease of explanation, we assume that $n = |V(G)|$ is even; the case where n is odd is similar. We first sketch Hassin and Rubinstein's algorithm (H&R-algorithm) for Max TSP in the next section, and then detail how to improve it in the subsequent sections.

3 H&R-algorithm

H&R-algorithm starts by computing a maximum-weight cycle cover $\mathcal{C}$. If $\mathcal{C}$ is a tour of G, then we are done. Throughout the rest of the paper, we assume that $\mathcal{C}$ is not a tour of G. Suppose that

T is a maximum-weight tour of G. Let $T_{\rm int}$ denote the set of all edges $\{u, v\}$ of T such that some cycle C in $\mathcal{C}$ contains both u and v. Let $T_{\rm ext}$ denote the set of edges in T but not in $T_{\rm int}$. Let $\alpha = w(T_{\rm int})/w(T)$.

H&R-algorithm then computes three tours T_1, T_2, T_3 of G and outputs the one of the largest weight. Based on an idea in [3], T_1 is computed by modifying the cycles in $\mathcal{C}$ as follows. Fix a parameter $\epsilon > 0$. For each cycle C in $\mathcal{C}$, if $|E(C)| > \epsilon^{-1}$, then remove the minimum-weight edge; otherwise, replace C by a maximum-weight path P in G with $V(P) = V(C)$. Then, $\mathcal{C}$ becomes a subtour and we can extend it to a tour T_1 in an arbitrary way. As observed by Hassin and Rubinstein [4], we have:

Fact 3.1 $w(T_1) \geq (1-\epsilon)w(T_{\rm int}) = (1-\epsilon)\alpha w(T)$.

When $w(T_{\rm ext})$ is large, $w(T_{\rm int})$ is small and $w(T_1)$ may be small, too. The two tours T_2 and T_3 together are aimed at the case where $w(T_{\rm ext})$ is large. By modifying Serdyukov's algorithm, T_2 and T_3 are computed as shown in Figure 1:

1. Compute a maximum-weight matching M in G. (Comment: Since $|V(G)|$ is even, M is perfect.)

2. Compute a maximum-weight matching M' in a graph H, where $V(H) = V(G)$ and $E(H)$ consists of those $\{u, v\} \in E(G)$ such that u and v belong to different cycles in $\mathcal{C}$.

3. Let $C_1, \ldots, C_r$ be an arbitrary ordering of the cycles in $\mathcal{C}$.

4. For $i = 1, 2, \ldots, r$ (in this order), perform the following steps:

- **(a)** Compute two disjoint nonempty matchings A_1 and A_2 in C_i such that each vertex of C_i is incident to an edge in $A_1 \cup A_2$ and both the graphs $(V(G), M \cup A_1)$ and $(V(G), M \cup A_2)$ are subtours of G.
- **(b)** Select $h \in \{1, 2\}$ uniformly at random, and move the edges in A_h from $\mathcal{C}$ to M.

5. Add those edges $\{u, v\} \in M'$ to $\mathcal{C}$ such that both u and v have degree 1 in $\mathcal{C}$.

6. For each cycle C in $\mathcal{C}$, select one edge in $E(C) \cap M'$ uniformly at random and delete it from C.

7. Complete $\mathcal{C}$ to a tour T_2 of G by adding some edges of G.

8. Complete the graph $(V(G), M)$ to a tour T_3 of G by adding some edges of G.

Figure 1. Computation of tours T_2 and T_3 in H&R-algorithm.

4 Ideas for Improving H&R-algorithm

A bottleneck of H&R-algorithm is that at the beginning of Step 6, there may exist many cycles C in $\mathcal{C}$ with $|E(C) \cap M'| = 2$. Let us call such cycles C *bad cycles*. Fix a bad cycle C. Observe that if we remove the two edges in $E(C) \cap M'$ from C, we are left with two paths P and Q such that some cycle C_j in the original cycle cover $\mathcal{C}$ contains P and another cycle C_k in the original cycle cover $\mathcal{C}$ contains Q, where $j > k$. When executing Step 4 with $i = j$, we know those pairs $(\{u_1, v_1\}, \{u_2, v_2\})$ of edges in M' such that $\{u_1, u_2\} \subseteq V(C_j)$, $\{v_1, v_2\} \subseteq V(C_k)$, and $\mathcal{C}$ has a path between v_1 and v_2. Let us call such pairs $(\{u_1, v_1\}, \{u_2, v_2\})$ of edges in M' *C_j-serious pairs*. Assume that when executing Step 4 with $i = j$, it were possible to compute two matchings A_1 and A_2 in C_j such that for each C_j-serious pair $(\{u_1, v_1\}, \{u_2, v_2\})$ and for each $h \in \{1, 2\}$, $C_j - A_h$

contains no path between u_1 and u_2. Then, we would have been able to avoid C. This is our main idea for improving H&R-algorithm.

Unfortunately, not all bad cycles can be avoided. Another idea in our algorithm is to discard a small-weighted subset R of edges in M' so that a large fraction of bad cycles can be avoided. In order to realize this idea, we need to choose a suitable ordering of the cycles in the original cycle cover $\mathcal{C}$ and process them in this order. In the course of processing the cycles, we will include some edges of M' into R. Yet another idea in our algorithm is to let the random selection at Step 6 in Figure 1 be sometimes nonuniform.

5 Outline of the New Algorithm

Like H&R-algorithm, our algorithm starts by computing a maximum-weight cycle cover $\mathcal{C}$ of G, uses it to compute three tours $T_1, \ldots, T_3$ of G, and outputs the one of the largest weight among them. Our computation of T_1 is the same as in H&R-algorithm. Our computation of T_2 and T_3 is as shown in Figure 2:

1. Perform Steps 1 and 2 in Figure 1 in turn.

2. Let $C_1, \ldots, C_r$ be an ordering of the cycles in $\mathcal{C}$ such that $C_1, \ldots, C_\ell$ are the 4-cycles in $\mathcal{C}$.

3. Make a backup copy M_c of M.

4. Process $C_1, \ldots, C_\ell$ in a suitable order, by (1) coloring some edges $\{u, v\} \in M'$ with $\{u, v\} \subseteq \cup_{1 \le i \le \ell} V(C_i)$ *red*, and (2) moving exactly one suitable edge from each 4-cycle to M while always maintaining that the graph $(V(G), M)$ is a subtour of G.

5. Process $C_{\ell+1}, \ldots, C_r$ one by one in this order, by (1) coloring some edges $\{u, v\} \in M'$ with $\{u, v\} \cap (\cup_{\ell+1 \le i \le r} V(C_i)) \ne \emptyset$ *red* or *green*, and (2) moving one or more suitable edges in each non-4-cycle to M while always maintaining that the graph $(V(G), M)$ is a subtour of G.

6. Add to $\mathcal{C}$ those edges $\{u, v\} \in M' - R$ such that both u and v have degree 1 in $\mathcal{C}$, where R is the set of *red* edges in M'. (Comment: Let M'_6 denote the set of edges in M' that are added to $\mathcal{C}$ at this step. Immediately after this step, $|E(C) \cap M'_6| \ge 2$ for each cycle C in $\mathcal{C}$.)

7. For each cycle C in $\mathcal{C}$, if $|E(C) \cap M'| = 2$ and one edge in $E(C) \cap M'$ is *green*, then delete one edge in $E(C) \cap M'$ from C at random in such a way that the *green* edge is deleted with probability $2/3$; otherwise, select one edge in $E(C) \cap M'$ uniformly at random and delete it from C. (Comment: Let M'_7 denote the set of edges in M' that remain in $\mathcal{C}$ immediately after this step.)

8. Perform Steps 7 and 8 in Figure 1 in turn.

Figure 2. Computation of T_2 and T_3 in our algorithm. (Steps 4 and 5 are rough.)

Steps 4 and 5 in Figure 2 are rough; their details are very complicated and will be given in the subsequent sections. An important property will be that $w(R)$ is small compared with $w(M')$.

Several definitions and three useful facts are in order. Throughout the rest of this paper, for each integer $i \in \{1, \ldots, r\}$, the phrase "*at time i*" means the time at which zero or more cycles in $\mathcal{C}$ have been processed and C_i is the next cycle to be processed. A set F of edges in G is *available at time i* if F is a matching in C_i, $F \cap M_c = \emptyset$, and the graph $(V(G), M \cup F)$ is a subtour of G at time i. An edge e in G is *available at time i* if $\{e\}$ is available at time i. A *maximal available set at time i* is an available set F at time i such that for every $e \in E(C_i) - F$, $F \cup \{e\}$ is not available at time i.

Lemma 5.1 *Let F be an available set at time i. Suppose that $e_1 = \{u_1, u_2\}$ and $e_2 = \{u_2, u_3\}$ are two adjacent edges in C_i such that F contains no edge incident to u_1, u_2, or u_3. Then, $F \cup \{e_1\}$ or $F \cup \{e_2\}$ is available at time i.*

Proof. If $F \cup \{e_1\}$ is available, then we are done. So, assume that $F \cup \{e_1\}$ is not available. Consider the subtour $H_{i,F} = (V(G), M \cup F)$ at time i. Since M_c is perfect and F contains no edge incident to u_1, u_2, or u_3, the degree of each $u_j \in \{u_1, u_2, u_3\}$ in $H_{i,F}$ is 1. In turn, since $F \cup \{e_1\}$ is not available, some connected component (a path) Q of $H_{i,F}$ is a path between u_1 and u_2 (no matter whether $e_1 \in M_c$ or not). Because a path can have at most two vertices of degree 1, u_3 is not a vertex of Q. So, $\{u_2, u_3\} \notin M_c$, and $H_{i,F}$ remains to be a subtour even after e_2 is added to it. Consequently, $F \cup \{e_2\}$ is available at time i. □

The following corollary is immediate from Lemma 5.1:

Corollary 5.2 *Suppose that F is a maximal available set at time i. Then, $C_i - F$ is a collection of vertex-disjoint paths each of length ≤ 3.*

Lemma 5.3 *Let F be an available set at time i. Suppose that $e_1 = \{u_1, u_2\}$, $e_2 = \{u_2, u_3\}$, $e_3 = \{u_3, u_4\}$, and $e_4 = \{u_4, u_5\}$ are four distinct (consecutive) edges in $E(C_i) - F$ such that no $u_i \in \{u_1, \ldots, u_5\}$ is incident to an edge in F and neither $F \cup \{e_1\}$ nor $F \cup \{e_3\}$ is available at time i. Then, $F \cup \{e_2, e_4\}$ is available at time i.*

Proof. As in the proof of Lemma 5.1, consider the subtour $H_{i,F} = (V(G), M \cup F)$ at time i. Since neither $F \cup \{e_1\}$ nor $F \cup \{e_3\}$ is available at time i, some connected component Q of $H_{i,F}$ is a path between u_1 and u_2, and another connected component Q' of $H_{i,F}$ is a path between u_3 and u_4. So, even after we add e_2 to $H_{i,F}$, u_4 and u_5 still belong to different connected components of $H_{i,F}$ and both u_4 and u_5 remain to have degree 1 in $H_{i,F}$. In turn, even after we add both e_2 and e_4 to $H_{i,F}$, $H_{i,F}$ still remains to be a subtour. In other words, $F \cup \{e_2, e_4\}$ is available at time i. □

6 Processing 4-Cycles

We say that two distinct edges $e_1 = \{u_1, v_1\}$ and $e_2 = \{u_2, v_2\}$ in M' form a *square pair*, denoted by $\{e_1, e_2\}_{\mathrm{sp}}$, if $\{u_1, u_2\}$ is an edge in a 4-cycle C_i and $\{v_1, v_2\}$ is an edge in another 4-cycle C_j. We call C_i and C_j the *dependent 4-cycles* of the square pair. An edge $e \in M'$ is a *square edge* if e is contained in some square pair.

We construct a multigraph H_1 from M' and $C_1, \ldots, C_\ell$ as follows. The nodes of H_1 one-to-one correspond to $C_1, \ldots, C_\ell$. For convenience, we still use C_i $(1 \leq i \leq \ell)$ to denote the node of H_1 corresponding to it. The edges of H_1 one-to-one correspond to the square pairs. In more detail, corresponding to each square pair p, H_1 has an edge between the dependent 4-cycles of p. H_1 has no other edges. For each edge f of H_1, we denote the square pair corresponding to f by $p(f)$.

An edge $\{u, v\} \in M'$ is *4-cycle-closed* if there are two 4-cycles C_i and C_j in $\mathcal{C}$ with $u \in V(C_i)$ and $v \in V(C_j)$. An edge $e \in M'$ is *4-cycle-pendent* if for exactly one endpoint u of e, there is a 4-cycle C_i in $\mathcal{C}$ with $u \in V(C_i)$. Let Q be a connected subgraph of H_1. An edge $\{u, v\} \in M'$ is *Q-closed* if there are two nodes C_i and C_j in Q with $u \in V(C_i)$ and $v \in V(C_j)$. An edge $e \in M'$ is *Q-pendent* if for exactly one endpoint u of e, there is a node C_i in Q with $u \in V(C_i)$. The *weight* of Q is the total weight of Q-closed edges in M', and is denoted by $w(Q)$.

Obviously, we can classify the connected components Q of H_1 into ten types as follows:

Type 1: Q is a single node.

Type 2: Q is a bunch of four parallel edges between two nodes.
Type 3: Q is an odd cycle.
Type 4: Q is an even cycle of length 4 or more.
Type 5: Q is a path of length 1 or more, and Q has an endpoint C_i (a 4-cycle in $\mathcal{C}$) such that neither a Q-pendent edge nor a Q-closed non-square edge is incident to a vertex of C_i. (Comment: We call C_i a *dead end* of Q. Note that if there is a Q-closed non-square edge, then Q has no dead end.)
Type 6: Q is a path of length 3 or more, and Q has no dead end.
Type 7: Q is a 2-cycle.
Type 8: Q is a path of length 1 and Q has no dead end.
Type 9: Q is a path of length 2, Q has no dead end, and there is no Q-closed non-square edge.
Type 10: Q is a path of length 2 and there is a Q-closed non-square edge.

The following two facts are obvious and help the reader understand the above definitions.

Fact 6.1 *Let M'_{4c} be the set of all 4-cycle-closed edges in M'. Let C be a cycle in the graph $(V(G), E(\mathcal{C}) \cup M'_{4c})$ with $|E(C) \cap M'| = 2$. Let e_1 and e_2 be the two edges in $E(C) \cap M'$. Let C_i be the 4-cycle containing an endpoint u_1 of e_1 and an endpoint u_2 of e_2. Let C_j be the 4-cycle containing the other endpoint v_1 of e_1 and the other endpoint v_2 of e_2. Suppose that the two edges e_1 and e_2 in $E(C) \cap M'$ do not form a square pair. Then, we cannot remove exactly one edge e_3 from C_i and exactly one edge e_4 from C_j so that each vertex in $\{u_1, v_1, u_2, v_2\}$ is an endpoint of e_3 or e_4.*

Fact 6.2 *Let Q be a connected component of H_1. Then, the following hold:*

1. *If Q is of Type-2, 3, or 4, then there is no Q-pendent edge in M' and every Q-closed edge in M' is a square edge.*

2. *If Q is of Type-5 or 7, then there are at most two Q-pendent edges in M' and every Q-closed edge in M' is a square edge.*

3. *Suppose that Q is of Type 6, 8, 9, or 10. Then, the following hold:*

   (a) *For each Q-pendent edge $\{u, v\}$, the node C_i of Q with $\{u, v\} \cap V(C_i) \neq \emptyset$ is an endpoint of Q.*

   (b) *There are at most four Q-pendent edges in M' and there is at most one Q-closed non-square edge in M'.*

   (c) *If there is a Q-closed non-square edge $\{u, v\}$ in M', then there are at most two Q-pendent edges in M' and the two 4-cycles containing u or v are the endpoints of Q.*

The following two simple results are very useful for processing 4-cycles.

Lemma 6.3 *Suppose that our algorithm has processed zero or more 4-cycles and that C_i and C_j are two distinct 4-cycles not yet processed. Let e_1 and e_2 be two nonadjacent edges in C_i such that for each $e_k \in \{e_1, e_2\}$, $e_k \notin M_c$ and the graph $(V(G), M \cup \{e_k\})$ is a subtour of G. Then, we can choose two nonadjacent edges e_3 and e_4 in $E(C_j) - M_c$ such that for each $e_x \in \{e_1, e_2\}$ and for each $e_y \in \{e_3, e_4\}$, the graph $(V(G), M \cup \{e_x, e_y\})$ is a subtour of G.*

Proof. Consider the graph $H_2 = (V(G), M \cup \{e_1, e_2\})$. The degree of each vertex in H_2 is at most 2 and the degree of each vertex of C_j in H_2 is 1. Moreover, if H_2 contains a cycle, then both e_1 and e_2 appear on the cycle. If C_j has no edge e such that $e \in M_c$ or adding e to H_2 creates a new cycle in H_2, then we can choose e_3 and e_4 to be any two nonadjacent edges in C_j. On the other hand, if C_j has an edge $e = \{v_1, v_2\}$ such that $e \in M_c$ or adding e to H_2 creates a new cycle in H_2, then we can choose e_3 and e_4 to be the two edges incident to exactly one of v_1 and v_2 because for each $e_y \in \{e_3, e_4\}$ adding e_y to H_2 does not create a new cycle in H_2. □

Corollary 6.4 *For every 4-cycle C_i in $\mathcal{C}$, there are two nonadjacent edges available at time i.*

Proof. The first 4-cycle C_i processed by the algorithm must have two nonadjacent edges available at time i. So, Lemma 6.3 implies this corollary. □

To process the 4-cycles in $\mathcal{C}$, our algorithm considers the connected components of H_1 one by one. When considering a connected component Q of H_1, our algorithm processes those 4-cycles (in a row) that are nodes of Q. Since the details heavily depend on the type of Q, we describe the Type-1 case immediately and describe the other cases in six separate subsections.

1. Let C_i be the node of Q (a 4-cycle of G).

2. Choose two nonadjacent edges e_1 and e_2 available at time i. (Comment: By Corollary 6.4, e_1 and e_2 exist.)

3. Select an $e'' \in \{e_1, e_2\}$ uniformly at random, and move e'' from C_i to M.

Figure 3. Steps for processing a Type-1 connected component Q of H_1.

In general, immediately after considering a connected component Q of H_1 and processing the 4-cycle(s) that are nodes of Q, the following three invariants hold:

(I1) The graph $(V(G), M)$ remains to be a subtour of G.

(I2) Let C_i be a 4-cycle that is a node of Q. Then, exactly one edge of C_i was moved from C_i to M during considering Q.

(I3) Let u be a vertex in a 4-cycle C_i that is a node of Q. Suppose that no Q-closed edge in M' is incident to u. Then, with probability at least $1/2$, exactly one edge of C_i incident to u was moved from C_i to M during considering Q.

Obviously, immediately after considering a Type-1 connected component Q of H_1, Invariants (I1) through (I3) hold.

6.1 Type-2, Type-3, or Type-4 Connected Components

Let Q be a Type-2, Type-3, or Type-4 connected component of H_1. Then, there is no Q-pendent edge in M', and there is no Q-closed non-square edge in M', either. So, immediately after considering Q, Invariant (I3) is trivially true. In detail, to process Q, our algorithm performs the following steps:

1. If Q is of Type-2, then perform the following steps:

   (a) Let C_i and C_j be the nodes of Q (4-cycles of G). Let $e_1, \ldots, e_4$ be the four edges in M' each of which has one endpoint in C_i and the other in C_j.

(b) Compute an edge $e \in \{e_1, \ldots, e_4\}$ such that $w(e) \geq w(e_x)$ for all $x \in \{1, \ldots, 4\}$.

(c) Color e *blue*. (Comment: The total weight of Q-closed edges in M' is at most 4 times the weight of the edge colored *blue* at this step.)

(d) Find an edge $e' \in E(C_i) - M_c$ incident to an endpoint of e such that the graph $(V(G), M \cup \{e'\})$ is a subtour of G; further move e' from C_i to M. (Comment: By Corollary 6.4, e' exists.)

(e) Find an edge $e'' \in E(C_j) - M_c$ incident to an endpoint of e such that the graph $(V(G), M \cup \{e''\})$ is a subtour of G; further move e'' from C_j to M. (Comment: By Corollary 6.4, e'' exists. Moreover, Invariants (I1) through (I3) still hold after this step.)

(f) Color all uncolored Q-closed edges *red*.

2. If Q is of Type-3, then perform the following steps:

(a) Find an edge f_1 of Q such that $\max_{e \in p(f_1)} w(e) \leq \max_{e \in p(f_2)} w(e)$ for all edges f_2 of Q.

(b) Partition $E(Q) - \{f_1\}$ into two disjoint matchings N_1 and N_2.

(c) Compute an integer $h \in \{1, 2\}$ such that $\sum_{f \in N_h} \max_{e \in p(f)} w(e) \geq \sum_{f \in N_{h'}} \max_{e \in p(f)} w(e)$, where h' is the integer in $\{1, 2\} - \{h\}$.

(d) For each edge $f \in N_h$, perform the following steps:

i. Let C_i and C_j be the dependent 4-cycles of $p(f)$.

ii. Compute an edge $e \in p(f)$ such that $w(e) = \max_{e' \in p(f)} w(e')$.

iii. Perform Steps 1c through 1e.

(Comment: The total weight of Q-closed edges in M' is at most 6 times the total weight of edges colored *blue* at Step 2(d)iii.)

(e) Color all uncolored Q-closed edges *red*.

(f) For each node C_i of Q incident to no edge in N_h, select an arbitrary edge $e'' \in E(C_i) - M_c$ such that the graph $(V(G), M \cup \{e''\})$ is a subtour of G; further move e'' from C_i to M. (Comment: By Corollary 6.4, e'' exists. Moreover, Invariants (I1) through (I3) still hold after this step.)

3. If Q is of Type-4, then perform the following steps:

(a) Partition $E(Q)$ into two disjoint matchings N_1 and N_2.

(b) Perform Steps 2c through 2e. (Comment: Invariants (I1) through (I3) still hold after this step.)

The following lemma should be clear from the comments on Step 1c and 2(d)iii:

Lemma 6.5 *Immediately after the above steps for Q, Invariants (I1) through (I3) and the following hold:*

1. *The total weight of Q-closed edges in M' is at most 6 times the total weight of* blue *Q-closed edges in M'.*

2. *Let $\mathcal{C}'$ be the graph obtained from $\mathcal{C}$ by adding all non-*red *edges in M'. Then, for each* blue *Q-closed edge in M', the degree of each endpoint of e in $\mathcal{C}'$ is at most 2 and no cycle in $\mathcal{C}'$ contains e.*

The following corollary follows from Lemma 6.5 immediately:

Corollary 6.6 *Recall M'_7 in the comment on Step 7 in Figure 2. Let S be the set of Q-closed edges in M'. Then, $\mathcal{E}[w(S \cap M'_7)] \geq w(S)/6$.*

6.2 Type-5 Connected Components

Let Q be a Type-5 connected component of H_1. In order to maintain Invariant (I3), we need to carefully deal with the endpoint of path Q that is not a dead end of Q. In detail, to process Q, our algorithm performs the following steps:

1. Let C_{i_1} and C_{i_2} be the endpoints of path Q, where node C_{i_2} is a dead end of Q. Let $f_1 = \{C_{i_1}, C_{i_3}\}$ be the edge of Q incident to node C_{i_1}. (Comment: If $|E(Q)| = 1$, then $i_3 = i_2$.)

2. Let E_{i_1} be a set of two nonadjacent edges in $E(C_{i_1}) - M_{\rm c}$ such that for each $e_x \in E_{i_1}$, the graph $(V(G), M \cup \{e_x\})$ is a subtour of G. (Comment: By Corollary 6.4, E_{i_1} exists.)

3. Partition $E(Q)$ into two disjoint matchings N_1 and N_2.

4. Select an $h \in \{1, 2\}$ uniformly at random.

5. If $f_1 \in N_h$, then perform the following steps:

   (a) Select an $e \in p(f_1)$ uniformly at random.
   (b) Color e *purple*, and color the other edge in $p(f_1)$ *red*.
   (c) Move the edge in E_{i_1} adjacent to e from C_{i_1} to M.
   (d) Find an edge $e' \in E(C_{i_3}) - M_{\rm c}$ adjacent to e such that the graph $(V(G), M \cup \{e'\})$ is a subtour of G; further move e' from C_{i_3} to M. (Comment: By Corollary 6.4, e' exists.)

6. If $f_1 \notin N_h$, then perform the following step:

   (a) If there is an edge $e' \in E_{i_1}$ such that no edge in $p(f_1)$ is adjacent to e', then move e' from C_{i_1} to M; otherwise, select an $e'' \in E_{i_1}$ uniformly at random, and move e'' from C_{i_1} to M.

   (Comment: Obviously, Invariant (I3) still holds after this step.)

7. If node C_{i_2} is incident to no edge in N_h, then move an edge $e \in E(C_{i_2}) - M_{\rm c}$ from C_{i_2} to M such that the graph $(V(G), M \cup \{e\})$ is a subtour of G. (Comment: By Corollary 6.4, e exists.)

8. For each edge $f \in N_h - \{f_1\}$, perform the following steps:

   (a) Let C_i and C_j be the dependent 4-cycles of $p(f)$.
   (b) Select an $e \in p(f)$ uniformly at random.
   (c) Color e *purple*, and color the other edge in $p(f)$ *red*.
   (d) Perform Steps 1d and 1e in Section 6.1.

   (Comment: After this step, Invariants (I1) and (I2) hold.)

9. Color all uncolored Q-closed edges *red*.

The following lemma should be clear:

Lemma 6.7 *Immediately after the above steps for Q, Invariants (I1) through (I3) and the following hold:*

1. *For each Q-closed square edge e in M', the probability that e was colored* purple *during considering Q is at least* $1/4$.

2. *Let $\mathcal{C}'$ be the graph obtained from $\mathcal{C}$ by adding all non-*red *edges in M'. Then, for each edge $e \in M'$ colored* purple *during considering Q, the degree of each endpoint of e in $\mathcal{C}'$ is at most 2 and no cycle in $\mathcal{C}'$ contains e.*

Since Q is of Type-5, there is no Q-closed non-square edge. So, the following corollary follows from Lemma 6.7 immediately:

Corollary 6.8 *Let S be the set of Q-closed edges in M'. Then,* $\mathcal{E}[w(S \cap M'_7)] \geq w(S)/4$.

6.3 Type-6 Connected Components

Let Q be a Type-6 connected component of H_1. In order to maintain Invariant (I3), we need to carefully deal with both endpoints of path Q. This is the difficulty. To overcome this difficulty, the idea is to delete a *light* edge f from Q and then apply the steps in Section 6.2 or the steps in Figure 3 to each connected component (a path) of Q. Here, the word "light" means that $\sum_{e\in p(f)} w(e)$ is the smallest among all edges of Q. Because f is light and Q was originally long (of length 3 or more), the weight of Q is at least two thirds of its original weight.

In detail, to process Q, our algorithm performs the following steps:

1. Find an edge f_1 of Q such that $\sum_{e\in p(f_1)} w(e) \leq \sum_{e\in p(f_2)} w(e)$ for all edges f_2 of Q.

   (Comment: We call the two edges in $p(f_1)$ Q-closed *sacrifice* edges. Note that the total weight of Q-closed square edges is at least three times the total weight of Q-closed sacrifice edges.)

2. Color the edges in $p(f_1)$ *red*.

3. Let Q_1 and Q_2 be the connected components (paths) of $Q - \{f_1\}$.

4. For each Q_h ($h \in \{1, 2\}$), if $|E(Q_h)| = 0$, then apply the steps in Figure 3 to Q_h (by replacing each occurrence of Q there with Q_h here); otherwise, apply the steps in Section 6.2 to Q_h (by replacing each occurrence of Q there with Q_h here).

5. If there is a Q-closed non-square edge $e = \{u, v\}$ in M', then perform the following step:

   (a) If both an edge of $\mathcal{C}$ incident to u and an edge of $\mathcal{C}$ incident to v were moved to M at Step 4, then color e *yellow*; otherwise, color e *red*.

The following lemma should be clear:

Lemma 6.9 *Immediately after the above steps for Q, Invariants (I1) through (I3) still holds, and Statements 1 and 2 in Lemma 6.7 hold here, too.*

Corollary 6.10 *Let S be the set of Q-closed edges in M'. Then,* $\mathcal{E}[w(S \cap M'_7)] \geq w(S)/6$.

Proof. Let S_1 be the set of Q-closed square edges in M'. Let S_2 be the set of Q-closed sacrifice edges. Then, by our choice of Q-closed sacrifice edges, $w(S_1 - S_2) \geq 2w(S_1)/3$. On the other hand, $\mathcal{E}[w((S_1 - S_2) \cap M'_7)] \geq w(S_1 - S_2)/4$ by Lemma 6.9. So, $\mathcal{E}[w((S_1 - S_2) \cap M'_7)] \geq w(S_1)/6$. It remains to prove that for every $e \in S - S_1$, $\Pr[e \in M'_7] \geq \frac{1}{6}$.

Suppose $e \in S - S_1$. By Lemma 6.7 and Invariant (I3), e is colored *yellow* at Step 5 above with probability at least $\frac{1}{4}$, and in turn $\Pr[e \in M'_6] \geq \frac{1}{4}$ (recall M'_6 in the comment on Step 6 in Figure 2). Moreover, $\Pr[e \in M'_7 \mid e \in M'_6] \geq \frac{2}{3}$, because Fact 6.1 guarantees that after Step 6 in Figure 2, no cycle C in $\mathcal{C}$ can satisfy both $e \in E(C)$ and $|E(C) \cap M'| = 2$. Hence, $\Pr[e \in M'_7] \geq \frac{1}{6}$. □

6.4 Type-7 or Type-8 Connected Components

Let Q be a Type-7 or Type-8 connected component of H_1. Again, in order to maintain Invariant (I3), we need to carefully deal with the two nodes of Q. Moreover, since Q has very few edges, we cannot afford to delete any edge of Q. Fortunately, since Q has only two nodes, things turn out to be easy. In detail, to process Q, our algorithm performs the following steps:

1. Let C_i and C_j be the nodes of Q (4-cycles of G).

2. If Q is of Type-7, then perform the following steps:

   (a) Let $\{\{u_1, v_1\}, \{u_2, v_2\}\}_{\rm sp}$ and $\{\{u_2, v_2\}, \{u_3, v_3\}\}_{\rm sp}$ be the square pairs corresponding to the edges of Q, where $\{u_1, u_2, u_3\} \subseteq V(C_i)$ and $\{v_1, v_2, v_3\} \subseteq V(C_j)$.

   (b) Let u_4 be the vertex in $V(C_i) - \{u_1, u_2, u_3\}$. Let v_4 be the vertex in $V(C_i) - \{v_1, v_2, v_3\}$.

   (c) Let e_1 and e_2 be two nonadjacent edges in $E(C_i) - M_{\rm c}$ such that the graphs $(V(G), M \cup \{e_1\})$ and $(V(G), M \cup \{e_2\})$ are subtours of G. (Comment: By Corollary 6.4, e_1 and e_2 exist.)

   (d) Choose two nonadjacent edges e_3 and e_4 of C_j as stated in Lemma 6.3.

   (e) Let Ω be the set of all (ordered) pairs (e_x, e_y) such that (1) $e_x \in \{e_1, e_2\}$, (2) $e_y \in \{e_3, e_4\}$, and (3) there is a $k \in \{1, 2, 3, 4\}$ such that both u_k and v_k are of degree 1 in the graph $\mathcal{C} - \{e_x, e_y\}$. (Comment: By a simple case-analysis where one looks at which edges of C_i are in $\{e_1, e_2\}$ and which edges of C_j are in $\{e_3, e_4\}$, we can prove that either $|\Omega| = 2$ or $|\Omega| = 4$.)

   (f) Select a pair (e_x, e_y) from Ω uniformly at random.

   (g) Move e_x from C_i to M, and move e_y from C_j to M.

   (Comment: After this step, Invariants (I1) through (I3) still hold. In particular, Invariant (I3) can be seen by a simple case-analysis where one looks at which edges of C_i are in $\{e_1, e_2\}$ and which edges of C_j are in $\{e_3, e_4\}$.)

   (h) If there is a unique $k \in \{1, 2, 3\}$ such that both u_k and v_k have just become of degree 1 in $\mathcal{C}$, then color edge $\{u_k, v_k\}$ *yellow*.

   (i) If there are two integers $k \in \{1, 2, 3\}$ such that both u_k and v_k have just become of degree 1 in $\mathcal{C}$, then select one of them uniformly at random and color it *yellow*.

   (Comment: For each Q-closed edge e, the probability that e is colored *yellow* at Step 2h or 2i is at least $1/4$. This can be seen by a simple case-analysis where one looks at which edges of C_i are in $\{e_1, e_2\}$ and which edges of C_j are in $\{e_3, e_4\}$.)

   (j) Color all uncolored Q-closed edges *red*.

3. If Q is of Type-8, then perform the following steps:

   (a) Let $\{\{u_1, v_1\}, \{u_2, v_2\}\}_{\rm sp}$ be the square pair corresponding to the edge of Q, where $\{u_1, u_2\} \subseteq V(C_i)$ and $\{v_1, v_2\} \subseteq V(C_j)$.

(b) Let u_3, u_4 be an ordering of the two nodes in $V(C_i) - \{u_1, u_2\}$ and v_3, v_4 be an ordering of the two nodes in $V(C_j) - \{v_1, v_2\}$ such that if there is a Q-closed non-square edge in M', then that edge is $\{u_3, v_3\}$. (Comment: See Statement 3b in Fact 6.2.)

(c) Perform Steps 2c through 2g.

(d) If there is a unique $k \in \{1, 2, 3\}$ such that both u_k and v_k have just become of degree 1 in C_i and $\{u_k, v_k\}$ is a Q-closed edge in M', then color edge $\{u_k, v_k\}$ *yellow*.

(e) If there are two integers $k \in \{1, 2, 3\}$ such that both u_k and v_k have just become of degree 1 in $\mathcal{C}$ and $\{u_k, v_k\}$ is a Q-closed edge in M', then select one of them uniformly at random and color it *yellow*.

(Comment: For each Q-closed edge e, the probability that e is colored *yellow* at Step 3d or 3e is at least $1/4$. This can be seen by a simple case-analysis where one looks at which edges of C_i are in $\{e_1, e_2\}$ and which edges of C_j are in $\{e_3, e_4\}$.)

(f) Color all uncolored Q-closed edges *red*.

The following should be clear from the comments on Steps 2i and 3e.

Lemma 6.11 *Immediately after the above steps for Q, Invariants (I1) through (I3) and the following hold:*

1. *For each Q-closed edge e in M', the probability that e was colored* yellow *during considering Q is at least* $1/4$.

2. *Let $\mathcal{C}'$ be the graph obtained from $\mathcal{C}$ by adding all non-*red *edges in M'. Then, for each edge $e \in M'$ colored* yellow *during considering Q, the degree of each endpoint of e in $\mathcal{C}'$ is at most 2 and each cycle in $\mathcal{C}'$ containing e contains at least three edges in M'.*

Corollary 6.12 *Let S be the set of Q-closed edges in M'. Then, $\mathcal{E}[w(S \cap M'_7)] \geq w(S)/6$.*

Proof. It suffices to prove that for every $e \in S$, $\Pr[e \in M'_7] \geq \frac{1}{6}$. Suppose $e \in S$. By Lemma 6.11, $\Pr[e \in M'_6] \geq \frac{1}{4}$. Moreover, $\Pr[e \in M'_7 \mid e \in M'_6] \geq \frac{2}{3}$ by Statement 2 in Lemma 6.11. Thus, $\Pr[e \in M'_7] \geq \frac{1}{6}$. □

6.5 Type-9 Connected Components

Let Q be a Type-9 connected component of H_1. Yet again, in order to maintain Invariant (I3), we need to carefully deal with the endpoints of Q. This is not difficult because there is no Q-closed non-square edge. In detail, to process Q, our algorithm performs the following steps:

1. Let C_{i_1} and C_{i_2} be the endpoints of path Q. Let E_{i_1} be a set of two nonadjacent edges in $E(C_{i_1}) - M_{\mathrm{c}}$ such that for each $e_x \in E_{i_1}$, the graph $(V(G), M \cup \{e_x\})$ is a subtour of G. (Comment: By Corollary 6.4, E_{i_1} exists.)

2. Choose a set E_{i_2} of two nonadjacent edges in $E(C_{i_2}) - M_{\mathrm{c}}$ such that for each $e_x \in E_{i_1}$ and for each $e_y \in E_{i_2}$, the graph $(V(G), M \cup \{e_x, e_y\})$ is a subtour of G. (Comment: By Lemma 6.3, E_{i_2} exists.)

3. Let C_{i_3} be the other node of Q than C_{i_1} and C_{i_2}.

4. Select a Q-closed square edge $e = \{u, v\}$ uniformly at random. (Comment: Since there are exactly four Q-closed square edges in M', each Q-closed square edge is selected at this step with probability 1/4.)

5. Color e *purple* and color the other Q-closed square edges in M' *red*.

6. Let i_4 be the integer in $\{i_1, i_2\}$ with $\{u, v\} \cap V(C_{i_4}) \neq \emptyset$. Let i_5 be the other integer in $\{i_1, i_2\} - \{i_4\}$.

7. If there is an edge $e' \in E_{i_5}$ adjacent to no Q-closed square edge in M', then move e' from C_{i_5} to M; otherwise, select an $e' \in E_{i_5}$ uniformly at random, and move e' from C_{i_5} to M.

8. Move the edge in E_{i_4} incident to u or v from C_{i_4} to M. (Comment: After this step, $(V(G), M)$ remains to be a subtour of G by our choice at Step 2.)

9. Move an edge e'' of C_{i_3} incident to u or v from C_{i_3} to M such that the graph $(V(G), M)$ remains to be a subtour of G.

   (Comment: By Corollary 6.4, e'' exists. Moreover, after this step, Invariants (I1) through (I3) still hold. In particular, Invariant (I3) can be seen by a simple case-analysis where one looks at which edges of C_{i_1} are in E_{i_1} and which edges of C_{i_2} are in E_{i_2}.)

The following lemma should be clear:

Lemma 6.13 *Immediately after the above steps for Q, Invariants (I1) through (I3) still holds, and Statements 1 and 2 in Lemma 6.7 hold here, too.*

Corollary 6.14 *Let S be the set of Q-closed edges in M'. Then, $\mathcal{E}[w(S \cap M'_7)] \geq w(S)/4$.*

6.6 Type-10 Connected Components

Let Q be a Type-10 connected component of H_1. Let e_1 be the unique Q-closed non-square edge in M'. Still again, in order to maintain Invariant (I3), we need to carefully deal with the endpoints of Q. This is not easy because of the existence of e_1. Fortunately, since there are only five Q-closed edges in M', things turn out to be easy. In detail, to process Q, our algorithm performs the following steps:

1. Perform Steps 1 through 3 in Section 6.5 in turn.

2. If $w(e_1) \leq w(Q)/3$, then color e_1 *red*; otherwise, find an edge f_1 of Q with $\sum_{e \in p(f_1)} w(e) \leq w(Q)/3$, and color the two edges in $p(f_1)$ *red*.

   (Comment: We call the edge(s) colored *red* at Step 2 Q-closed *sacrifice* edge(s).)

3. If e_1 is *red*, then perform Steps 4 through 9 in Section 6.5 in turn. (Comment: See Lemma 6.13.)

4. If e_1 is uncolored and E_{i_1} or E_{i_2} contains an edge adjacent to a Q-closed square edge in M', then perform Steps 3 through 5 in Section 6.3 in turn. (Comment: See Lemma 6.9.)

5. If e_1 is uncolored and both E_{i_1} and E_{i_2} contain an edge adjacent to no Q-closed square edge in M', then perform the following steps:

   (a) Select an uncolored Q-closed edge e_2 at random in such a way that $e_2 = e_1$ with probability 1/2 and each of the two uncolored Q-square edges is e_2 with probability 1/4.

(b) If $e_2 = e_1$, then color e_1 *orange*, move the two edges in $E_{i_1} \cup E_{i_2}$ adjacent to e_1 from $\mathcal{C}$ to M, and move an edge e' of C_{i_3} from $\mathcal{C}$ to M such that the graph $(V(G), M)$ remains to be a subtour of G. (Comment: By Corollary 6.4, e' exists.)

(c) If $e_2 \neq e_1$, then color e_2 *purple*, move the two edges in $E_{i_1} \cup E_{i_2}$ not adjacent to e_1 from $\mathcal{C}$ to M, and move an edge e' of C_{i_3} from $\mathcal{C}$ to M such that the graph $(V(G), M)$ remains to be a subtour of G. (Comment: By Corollary 6.4, e' exists.)

(d) Color all uncolored edges *red*.

The following lemma should be clear:

Lemma 6.15 *Immediately after the above steps for Q, Invariants (I1) through (I3) still holds, and Statements 1 and 2 in Lemma 6.7 hold here, too.*

Corollary 6.16 *Let S be the set of Q-closed edges in M'. Then, $\mathcal{E}[w(S \cap M'_7)] \geq w(S)/6$.*

PROOF. Let S_1 be the set of Q-closed square edges in M'. Let S_2 be the set of Q-closed sacrifice edges. Then, by our choice of Q-closed sacrifice edges, $w(S - S_2) \geq 2w(S)/3$. Thus, it suffices to prove that for each edge $e \in S - S_2$, $\Pr[e \in M'_7] \geq \frac{1}{4}$. By Lemma 6.15, $\Pr[e \in M'_7] \geq \frac{1}{4}$ for every $e \in S_1 - S_2$. So, it remains to prove that if the unique Q-closed non-square edge e_1 is in $S - S_2$, then $\Pr[e_1 \in M'_7] \geq \frac{1}{4}$.

Suppose $e_1 \in S - S_2$. Then, either the condition in Step 4 or the condition in Step 5 is true. We distinguish two cases as follows:

Case 1: The condition in Step 4 is true. Then, as observed in the proof of Corollary 6.10, e_1 is colored *yellow* at Step 4 (more precisely at Step 5 in Section 6.3) with probability at least $\frac{1}{4}$, and hence $\Pr[e_1 \in M'_6] \geq \frac{1}{4}$. A crucial observation is that if the event $e_1 \in M'_6$ occurs, then after Step 6 in Figure 2, no cycle of $\mathcal{C}$ contains e_1 and so $e_1 \in M'_7$. This can be seen from the condition in Step 4 by a simple case-analysis where one looks at which edges of C_{i_1} are in E_{i_1} and which edges of C_{i_2} are in E_{i_2}. Thus, $\Pr[e_1 \in M'_7 \mid e_1 \in M'_6] = 1$ and in turn $\Pr[e_1 \in M'_7] \geq \frac{1}{4}$.

Case 2: The condition in Step 5 is true. Then, by Steps 5a and 5b, e_1 is colored *orange* at Step 5b with probability $\frac{1}{2}$, and in turn $\Pr[e_1 \in M'_6] \geq \frac{1}{2}$. Moreover, if the event $e_1 \in M'_6$ occurs, then after Step 6 in Figure 2, each cycle C in $\mathcal{C}$ with $e \in E(C)$ satisfies $|E(C) \cap M'| \geq 3$ by Fact 6.1. This implies that $\Pr[e_1 \in M'_7 \mid e_1 \in M'_6] \geq \frac{2}{3}$. Thus, $\Pr[e_1 \in M'_7] \geq \frac{1}{4}$ in this case, too. □

6.7 A Main Lemma

We are now ready to prove the following:

Lemma 6.17 *Immediately after Step 4 in Figure 2 (i.e., immediately after processing the 4-cycles $C_1, \ldots, C_\ell$), the following hold:*

1. *The graph $(V(G), M)$ is a subtour of G.*

2. *Each C_i $(1 \leq i \leq \ell)$ becomes a path in $\mathcal{C}$.*

3. *Let e be a 4-cycle-pendent edge in M'. Then, with probability at least $1/2$, the endpoint of e in a C_i $(1 \leq i \leq \ell)$ is of degree 1 in $\mathcal{C}$.*

4. *Let S be the set of 4-cycle-closed edges in M'. Then, $\mathcal{E}[w(S \cap M'_7)] \geq w(S)/6$.*

Proof. The first three statements are obviously true from Invariants (I1) through (I3) and the lemmas in Sections 6.1 through 6.6. To see Statement 4, let S_1 be the set of Q-closed edges where Q ranges over all connected components of H_1. Then, by the corollaries in Sections 6.1 through 6.6, $\mathcal{E}[w(S_1 \cap M'_7)] \geq w(S_1)/6$. So, it suffices to show that for each $e \in S - S_1$, $\Pr[e \in M'_7] \geq \frac{1}{6}$.

Suppose that $e = \{u, v\}$ is an edge in $S - S_1$. Then, with probability at least $\frac{1}{4}$, both u and v are of degree 1 in $\mathcal{C}$ immediately after Step 4 in Figure 2. This follows from Invariant (I3) and the lemmas in Sections 6.1 through 6.6 immediately. So, $\Pr[e \in M'_6] \geq \frac{1}{4}$. Moreover, if the event $e \in M'_6$ occurs, then after Step 6 in Figure 2, each cycle C in $\mathcal{C}$ with $e \in E(C)$ satisfies $|E(C) \cap M'| \geq 3$ by Fact 6.1. This implies that $\Pr[e \in M'_7 \mid e \in M'_6] \geq \frac{2}{3}$. Thus, $\Pr[e \in M'_7] \geq \frac{1}{6}$. □

7 Ideas for Processing Non-4-Cycles

For convenience, we transform each edge $\{u, v\} \in M'$ to an ordered pair (u, v), where the C_i with $u \in V(C_i)$ and the C_j with $v \in V(C_j)$ satisfy that $i > j$.

Let i be an integer in $\{\ell+1, \ldots, r\}$. A *C_i-settled edge* is an edge $(u, v) \in M'$ such that $u \in V(C_i)$ (and so $v \in V(C_j)$ for some $j < i$). A C_i-settled edge (u, v) is *active* at time i if the degree of v in $\mathcal{C}$ at time i is 1. A *C_i-settled vertex* is a vertex of C_i incident to a C_i-settled edge.

A *matching-pair* in C_i is an (unordered) pair $\{A_1, A_2\}$ such that both A_1 and A_2 are (possibly empty) matchings in C_i. An *available matching-pair at time i* is a matching-pair $\{A_1, A_2\}$ in C_i such that both A_1 and A_2 are available at time i. A *maximal available matching-pair at time i* is a matching-pair $\{A_1, A_2\}$ in C_i such that both A_1 and A_2 are maximal available sets at time i.

To break the cycles $C_{\ell+1}, \ldots, C_r$, our algorithm processes them in this order. While processing C_i, our algorithm colors zero or more C_i-settled vertices *red* and computes an available matching-pair $\{A_1, A_2\}$ at time i satisfying the following two conditions (and additionally some other conditions to be specified later):

(C1) Both A_1 and A_2 are nonempty.

(C2) Each non-*red* vertex of C_i is incident to at least one edge in $A_1 \cup A_2$.

The details of coloring C_i-settled vertices and computing $\{A_1, A_2\}$ will be given later. After computing $\{A_1, A_2\}$, our algorithm then selects an integer $h \in \{1, 2\}$ uniformly at random, and move the edges of A_h from C_i to M. Since we only color some C_i-settled vertices *red* and move the edges of A_h to M while processing C_i, we indeed maintain the following invariants:

(I4) At time i ($\ell + 1 \leq i \leq r$), $M - M_{\mathrm{c}}$ consists of some edges of $C_1, \ldots, C_{i-1}$, and the graph $(V(G), M)$ is a subtour of G.

(I5) For each $i \in \{\ell + 1, \ldots, r\}$ and for each C_i-settled edge e, the probability that e is active at time i is at least $1/2$.

Note that Invariants (I4) and (I5) hold at time $\ell + 1$ by Lemma 6.17.

7.1 Serious Pairs, Critical Pairs, and Dangerous Pairs

Throughout this subsection, fix a C_i with $\ell + 1 \leq i \leq r$.

A *serious pair at time i* is an unordered pair $\{(u_1, v_1), (u_2, v_2)\}$ of C_i-settled edges satisfying the following condition:

- At time i, some connected component of $\mathcal{C}$ is a path between v_1 and v_2. (Comment: By this condition, both (u_1, v_1) and (u_2, v_2) are active at time i.)

Obviously, no edge in M' is contained in two or more serious pairs at time i.

A matching-pair $\{A_1, A_2\}$ in C_i *covers* a vertex u of C_i if at least one edge in $A_1 \cup A_2$ is incident to u. A matching-pair $\{A_1, A_2\}$ in C_i *favors* a vertex u of C_i if A_1 contains an edge $e_1 \in E(C_i)$ incident to u and A_2 contains an edge $e_2 \in E(C_i)$ incident to u (possibly $e_1 = e_2$). A matching-pair $\{A_1, A_2\}$ in C_i is *good* for a serious pair $p = \{(u_1, v_1), (u_2, v_2)\}$ at time i if $\{A_1, A_2\}$ satisfies at least one of the following three conditions:

(G1) For each $h \in \{1, 2\}$, $C_i - A_h$ has no path from u_1 to u_2 or at least one of u_1 and u_2 has degree 2 in $C_i - A_h$.

(G2) $\{A_1, A_2\}$ favors both u_1 and u_2.

(Comment: If Condition (G1) or (G2) is satisfied, we say that $\{A_1, A_2\}$ is *strongly good* for p.)

(G3) $\{A_1, A_2\}$ favors exactly one of u_1 and u_2.

(Comment: If this condition is satisfied but Condition (G1) is not, we say that $\{A_1, A_2\}$ is *weakly good* for p.)

A *critical pair at time i* is a serious pair $p = \{(u_1, v_1), (u_2, v_2)\}$ at time i such that there is a path Q from u_1 to u_2 in C_i with $|E(Q)| \leq 3$. We call the path Q a *witness path* of the critical pair p. Obviously, if $|E(C_i)| \geq 5$ and $\{u_1, u_2\} \in E(C_i)$, then Q is unique. Similarly, if $|E(C_i)| \geq 7$, then Q is unique. Hereafter, when $|E(C_i)| \geq 5$ and p is a critical pair at time i with a unique witness path, we will use the following notations and definitions:

- $Q(p)$ denotes the unique witness path of p.
- $\tilde{Q}(p)$ denotes the *extended witness path* of p, i.e., the path obtained from $Q(p)$ by adding the two edges in $E(C_i) - E(Q(p))$ incident to an endpoint of $Q(p)$.
- If $|E(Q(p))| = 1$, then a matching-pair $\{A_1, A_2\}$ in C_i is *harmful* for p if either $A_1 \cap E(\tilde{Q}(p)) = \{e_1, e_2\}$ and $A_2 \cap E(\tilde{Q}(p)) = \emptyset$, or $A_2 \cap E(\tilde{Q}(p)) = \{e_1, e_2\}$ and $A_1 \cap E(\tilde{Q}(p)) = \emptyset$, where e_1 and e_2 are the two edges in $E(\tilde{Q}(p)) - E(Q(p))$.
- If $|E(Q(p))| = 3$, then a matching-pair $\{A_1, A_2\}$ in C_i is *harmful* for p if either $A_1 \cap E(\tilde{Q}(p)) = \{e_1, e_2\}$ and $A_2 \cap E(\tilde{Q}(p)) = \{e_3\}$, or $A_2 \cap E(\tilde{Q}(p)) = \{e_1, e_2\}$ and $A_1 \cap E(\tilde{Q}(p)) = \{e_3\}$, where e_1 and e_2 are the two edges in $E(\tilde{Q}(p)) - E(Q(p))$ and e_3 is the edge of $Q(p)$ not incident to an endpoint of $Q(p)$.

Lemma 7.1 *An arbitrary maximal available matching-pair at time i is strongly good for every serious but not critical pair at time i.*

Proof. Immediate from Corollary 5.2 and Condition (G1). □

Using Corollary 5.2, we can prove the following lemma by a simple case-analysis:

Lemma 7.2 *Suppose that either $|E(C_i)| \geq 5$ and $p = \{(u_1, v_1), (u_2, v_2)\}$ is a critical pair at time i with $\{u_1, u_2\} \in E(C_i)$, or $|E(C_i)| \geq 7$ and $p = \{(u_1, v_1), (u_2, v_2)\}$ is a critical pair at time i with $\{u_1, u_2\} \notin E(C_i)$. Then, the following hold:*

1. *Suppose that a maximal available matching-pair $\{A_1, A_2\}$ at time i covers all vertices of $Q(p)$ but is not good for p. Then, $\{A_1, A_2\}$ is harmful for p.*

2. *If a matching-pair $\{A_1, A_2\}$ in C_i covers all vertices of $Q(p)$ and is not harmful for p, then every matching-pair $\{B_1, B_2\}$ in C_i with $A_1 \subseteq B_1$ and $A_2 \subseteq B_2$ covers all vertices of $Q(p)$ and is not harmful for p.*

Lemma 7.3 *Let p be a critical pair at time i having a witness path Q with $|E(Q)| = 2$. Let $\{A_1, A_2\}$ be a matching-pair in C_i covering all vertices of Q. Then, $\{A_1, A_2\}$ is good for p.*

PROOF. The lemma is obvious if $|E(C_i)| = 3$. So, suppose $|E(C_i)| \geq 5$. If $\{A_1, A_2\}$ satisfies Condition (G1) for p, then we are done. So, assume that $\{A_1, A_2\}$ does not satisfy Condition (G1) for p. Then, for some $h \in \{1, 2\}$, A_h contains both the edge in $E(C_i) - E(Q)$ incident to u_1 and the edge in $E(C_i) - E(Q)$ incident to u_2. Moreover, since $\{A_1, A_2\}$ covers all vertices of Q, one edge in Q is in $A_1 \cup A_2$. Thus, $\{A_1, A_2\}$ is weakly good for p. □

Lemma 7.4 *Suppose that $|E(C_i)| \geq 5$. Let $p = \{(u_1, v_1), (u_2, v_2)\}$ be a critical pair at time i having no witness path Q satisfying the following condition:*

(C3) *$\{e_1, e_2\}$ is an available set at time i, where e_1 and e_2 are the two edges in $E(C_i) - E(Q)$ incident to an endpoint of Q.*

Let $\{A_1, A_2\}$ be a maximal available matching-pair at time i covering all vertices of Q. Then, $\{A_1, A_2\}$ is good for p.

PROOF. By Lemma 7.3, we may assume that p has no witness path of length 2. In turn, if $\{u_1, u_2\} \in E(C_i)$ or $|E(C_i)| \geq 7$, then $\{A_1, A_2\}$ cannot be harmful for p (because the unique witness path Q of p does not satisfy Condition (C3)), and hence is good for p by Lemmas 7.2. So, it suffices to consider only the case where $|E(C_i)| = 6$ and the distance between u_1 and u_2 in C_i is 3. In this case, one can easily verify that $\{A_1, A_2\}$ is good for p. □

Because of Lemmas 7.3 and 7.4, we define a *dangerous pair at time i* to be a critical pair at time i that has a witness path Q of length 1 or 3 satisfying Condition (C3). A *dangerous edge at time i* is a C_i-settled edge contained in a dangerous pair at time i. A *dangerous vertex at time i* is a vertex u of C_i incident to a dangerous edge at time i.

7.2 A Useful Procedure

Figure 4 shows a procedure useful for computing an available matching-pair at time i that covers the vertices of a given subgraph P of C_i. In most cases, we call $FindMatch(i, \emptyset, \emptyset, C_i, e)$, where e is an available edge at time i.

Procedure $\mathit{FindMatch}(i, Y_1, Y_2, P, e)$

Input: An integer $i \in \{\ell{+}1, \ldots, r\}$; an available matching-pair $\{Y_1, Y_2\}$ at time i with $Y_1 \cap Y_2 = \emptyset$; a subgraph P of C_i and an edge e of P such that $|E(P)| \geq 2$, $E(P) \cap Y_1 = E(P) \cap Y_2 = \emptyset$, $Y_1 \cup \{e_1\}$ is available at time i, and either $P = C_i$ or P is a path in C_i beginning with e.

1. Let $e_1, \ldots, e_t$ be the edges in P (appearing in P in this order) where $e_1 = e$. Let u_1 be the endpoint of e_1 not incident to e_2. Let u_2 be the endpoint of e_t not incident to e_{t-1}. (Comment: If $P = C_i$, then $u_1 = u_2$.)

2. Initialize $Z_1 = Y_1$, $Z_2 = Y_2$, and $j = h = 1$.

3. While $j < t$, perform the following two steps:

- **(a)** If $Z_h \cup \{e_j\}$ is available at time i, then add e_j to Z_h and further increase j by 1; otherwise, add e_{j+1} to Z_h and further increase j by 2. (Comment: Immediately after this step, if either $j <= t$, or $j = t + 1$ but only one edge in Z_h is incident to u_2, then by Lemma 5.1, both Z_1 and Z_2 are available at time i.)
- **(b)** If $h = 1$, then set $h = 2$; otherwise, set $h = 1$.

4. If some Z_k with $k \in \{1, 2\}$ contains both edges of C_i incident to u_2, then ($e_{t-1} \notin Z_1 \cup Z_2$ and so) perform the following two steps:

- **(a)** Let e' be the edge in $E(C_i) - \{e_t\}$ incident to u_2. Let e'' be the edge in $E(C_i) - \{e_t, e'\}$ adjacent to e'.
- **(b)** If $e'' \in Z_1 \cup Z_2$, then delete e' from Z_k.
- **(c)** If $e'' \notin Z_1 \cup Z_2$, then move a suitable edge in $\{e_t, e'\}$ from Z_k to $Z_{k'}$ while maintaining that $Z_{k'}$ is available at time i, where k' is the integer in $\{1, 2\} - \{k\}$. (Comment: By Lemma 5.1, this step can be done.)

Output: The ordered pair (Z_1, Z_2).

Figure 4. A procedure useful for computing A_1 and A_2.

Lemma 7.5 *For the output (Z_1, Z_2) of $\mathit{FindMatch}(i, Y_1, Y_2, P, e)$, the following hold:*

1. *$\{Z_1, Z_2\}$ is an available matching-pair at time i with $Z_1 \cap Z_2 = \emptyset$.*
2. *$\{Z_1, Z_2\}$ covers all vertices in $V(P) - \{u_2\}$.*
3. *$\{Z_1, Z_2\}$ does not cover u_2 only if P is a path and $\{Y_1, Y_2\}$ does not cover u_2.*
4. *If $P = C_i$ or $|E(P)| \geq 4$, then neither $Z_1 - Y_1$ nor $Z_2 - Y_2$ is empty.*
5. *If $P \neq C_i$ (i.e., P is a path), then $e \in Z_1$.*
6. *For every $h \in \{1, 2\}$, if $Y_h - Z_h \neq \emptyset$, then $Y_h - Z_h$ consists of only the edge $e' \in E(C_i) - E(P)$ incident to u_2 and $\{Z_1, Z_2\}$ covers the other endpoint of e' than u_2.*
7. *If $P \neq C_i$, then there is no $j \in \{1, 2, \ldots, t - 2 - b\}$ such that for some $h \in \{1, 2\}$, $Z_h \cap \{e_j, e_{j+1}, e_{j+2}\} = \{e_j, e_{j+2}\}$ and $Z_{h'} \cap \{e_j, e_{j+1}, e_{j+2}\} = \emptyset$, where h' is the integer in $\{1, 2\} - \{h\}$, $b = 0$ if the edge $e' \in E(C_i) - E(P)$ incident to u_2 does not belong to $Y_1 \cup Y_2$, and $b = 1$ if $e' \in Y_1 \cup Y_2$.*
8. *If $P = C_i$ and $|E(C_i)| \geq 5$, then there is no $j \in \{2, 3, \ldots, t - 3\}$ such that for some $h \in \{1, 2\}$, $Z_h \cap \{e_j, e_{j+1}, e_{j+2}\} = \{e_j, e_{j+2}\}$ and $Z_{h'} \cap \{e_j, e_{j+1}, e_{j+2}\} = \emptyset$, where h' is the integer in $\{1, 2\} - \{h\}$.*

9. *If $|E(P)| \geq 6$, then it is impossible that $Z_1 \cap \{e_1, \ldots, e_5\} = \{e_3\}$ and $Z_2 \cap \{e_1, \ldots, e_5\} = \{e_1, e_5\}$.*

10. *If $P = C_i$, $|E(C_i)| \geq 6$, and e_4 is available at time i, then it is impossible that $Z_1 \cap \{e_1, \ldots, e_5\} = \{e_1, e_5\}$ and $Z_2 \cap \{e_1, \ldots, e_5\} = \{e_3\}$.*

Proof. The first six statements are obvious. Statements 7 and 8 follows from Step 3b immediately.

To see Statement 9, suppose that $|E(P)| \geq 6$. For a contradiction, assume that $Z_1 \cap \{e_1, \ldots, e_5\} = \{e_3\}$ and $Z_2 \cap \{e_1, \ldots, e_5\} = \{e_1, e_5\}$. Then, since e_1 was put to Z_1 at Step 3a, it holds that at the beginning of Step 4, $Z_1 \cap \{e_1, \ldots, e_3\} = \{e_1, e_3\}$ and $Z_2 \cap \{e_1, \ldots, e_3\} = \emptyset$, which is impossible by Step 3b.

To see Statement 10, suppose that $P = C_i$ and $|E(C_i)| \geq 6$. For a contradiction, assume that $Z_1 \cap \{e_1, \ldots, e_5\} = \{e_1, e_5\}$ and $Z_2 \cap \{e_1, \ldots, e_5\} = \{e_3\}$. Since $P = C_i$, we have $Y_1 = Y_2 = \emptyset$. So, by procedure *FindMatch*, neither $\{e_2\}$ nor $\{e_1, e_4\}$ is available at time i. For each e_j with $1 \leq j \leq 4$, let v_j and v_{j+1} be the endpoints of e_j. Let H_3 be the graph $(V(G), M)$ at time i. Note that the degree of each v_j ($1 \leq j \leq 5$) in H_3 is 1. Since $\{e_2\}$ is not available at time i, some connected component of H_3 is a path between v_2 and v_3. On the other hand, since both e_1 and e_4 are available at time i but $\{e_1, e_4\}$ is not, some connected component of H_3 is a path between v_2 and some $v_j \in \{v_4, v_5\}$. This leads to a contradiction because no path can have three endpoints. □

8 Details of Processing Non-4-Cycles

If there is no dangerous pair at time i, then our algorithm colors no vertex of C_i *red* and processes C_i as shown in Figure 5.

1. Find an available edge e at time i, and let (A_1, A_2) be the output of $FindMatch(i, \emptyset, \emptyset, C_i, e)$. (Comment: By Lemma 5.1, e exists. Moreover, by the first four statements in Lemma 7.5, $\{A_1, A_2\}$ is a matching-pair satisfying Conditions (C1) and (C2).)

2. Extend $\{A_1, A_2\}$ to a maximal available matching-pair at time i.

3. For each critical pair $\{(u_1, v_1), (u_2, v_2)\}$ at time i for which $\{A_1, A_2\}$ is weakly good, if $\{A_1, A_2\}$ favors u_1, then color (u_1, v_1) *green* and color (u_2, v_2) *black*; otherwise, color (u_1, v_1) *black* and color (u_2, v_2) *green*.

4. Select an $h \in \{1, 2\}$ uniformly at random.

5. Move the edges in A_h from C_i to M.

Figure 5. Processing C_i when there is no dangerous pair at time i.

Lemma 8.1 *Let S be the set of C_i-settled edges. Suppose that there is no dangerous pair at time i and our algorithm processes C_i as shown in Figure 5. Recall M'_7 in the comment on Step 7 in Figure 2. Then, $\mathcal{E}[w(S \cap M'_7)] \geq w(S)/6$.*

Proof. Let S_1 be the set of edges in S that are active at time i. Let S_2 be the set of edges in S_1 that are serious at time i. Consider an edge $e = (u, v)$ in S. By Invariant (I5), $\Pr[e \in S_1] \geq \frac{1}{2}$. So, it remains to show that $\Pr[e \in M'_7 \mid e \in S_1] \geq \frac{1}{3}$. To this end, we distinguish four cases as follows:

Case 1: $e \in S_1 - S_2$. In this case, the event $e \in M'_6$ occurs with probability at least $\frac{1}{2}$ (recall M'_6 in the comment on Step 6 in Figure 2). Moreover, if the event $e \in M'_6$ occurs, then after Step 6 in Figure 2, each cycle C in $\mathcal{C}$ with $e \in E(C)$ satisfies $|E(C) \cap M'| \geq 3$. This implies that $\Pr[e \in M'_7 \mid e \in M'_6] \geq \frac{2}{3}$. Thus, $\Pr[e \in M'_7 \mid e \in S_1 - S_2] \geq \frac{1}{3}$.

Case 2: $e \in S_2$. Let D_1 be the event that $\{A_1, A_2\}$ satisfies Condition (G1) for the serious pair $p = \{e, e'\}$ at time i containing e. Let D_2 be the event that $\{A_1, A_2\}$ satisfies Condition (G2) for p. Let D_3 be the event that $\{A_1, A_2\}$ satisfies Condition (G3) for p. By Lemmas 7.1, 7.3, and 7.4, at least one of the following three cases occurs:

Case 2.1: Event D_1 occurs. In this case, the event $e \in M'_6$ occurs with probability $\frac{1}{2}$. Moreover, if the event $e \in M'_6$ occurs, then after Step 6 in Figure 2, each cycle C in $\mathcal{C}$ with $e \in E(C)$ satisfies $|E(C) \cap M'| \geq 4$. This implies that $\Pr[e \in M'_7 \mid e \in M'_6] \geq \frac{3}{4}$. Thus, $\Pr[e \in M'_7 \mid e \in S_2 \,\wedge\, D_1] \geq \frac{3}{8}$.

Case 2.2: Event D_2 occurs. In this case, the event $e \in M'_6$ occurs with probability 1. Obviously, $\Pr[e \in M'_7 \mid e \in M'_6] \geq \frac{1}{2}$. So, $\Pr[e \in M'_7 \mid e \in S_2 \,\wedge\, D_2] \geq \frac{1}{2}$.

Case 2.3: Event D_3 occurs. In this case, the probability that $e \in M'_6$ occurs depends on whether $\{A_1, A_2\}$ favors u or not. If $\{A_1, A_2\}$ favors u, then the event $e \in M'_6$ always occurs and $\Pr[e \in M'_7 \mid e \in M'_6] = \frac{1}{3}$ (because e is colored *green* at Step 3 in Figure 5). On the other hand, if $\{A_1, A_2\}$ does not favor u, then the event $e \in M'_6$ occurs with probability 1/2 and $\Pr[e \in M'_7 \mid e \in M'_6] = \frac{2}{3}$ (because e is colored *black* at Step 3 in Figure 5). Thus, no matter whether $\{A_1, A_2\}$ favors u or not, $\Pr[e \in M'_7 \mid e \in S_2 \,\wedge\, D_3] \geq \frac{1}{3}$.

By Cases 2.1 through 2.3, we always have $\Pr[e \in M'_7 \mid e \in S_2] \geq 1/3$. Combining this with the result in Case 1, we now have $\Pr[e \in M'_7 \mid e \in S_1] \geq \frac{1}{3}$. □

Hereafter, we assume that there are at least one dangerous pair at time i. Then, $|E(C_i)| \geq 5$.

8.1 The Case with Only One Dangerous Pair

In this case, our algorithm colors no vertex of C_i *red*. Let $p = \{(u_1, v_1), (u_2, v_2)\}$ be the dangerous pair at time i. Note that if $|E(C_i)| = 5$, then $\{u_1, u_2\} \in E(C_i)$.

Lemma 8.2 *Suppose that $\{u_1, u_2\} \in E(C_i)$. Then, the following hold:*

1. *If $|E(C_i)| = 5$, then we can choose an ordering $e_1, \ldots, e_t$ of the edges in C_i (appearing in C_i in this order) such that e_1 is available at time i and $\{u_1, u_2\} = e_3$.*

2. *If $|E(C_i)| \geq 6$, then we can choose an ordering $e_1, \ldots, e_t$ of the edges in C_i (appearing in C_i in this order) such that e_1 is available at time i and $\{u_1, u_2\} = e_j$ for some $j \in \{3, 4\}$.*

3. *Let (Z_1, Z_2) be the output of $FindMatch(i, \emptyset, \emptyset, C_i, e_1)$ where the ordering $e_1, \ldots, e_t$ in Statement 1 or 2 of this lemma is used at Step 1 in Figure 4. Then, an arbitrary maximal available matching-pair $\{A_1, A_2\}$ at time i with $Z_1 \subseteq A_1$ and $Z_2 \subseteq A_2$ satisfies Conditions (C1) and (C2) and is good for p.*

Proof. Statements 1 and 2 follow from Lemma 5.1. To see Statement 3, first observe that $\{A_1, A_2\}$ satisfies Conditions (C1) and (C2) by the first four statements in Lemma 7.5. Now, by Statement 1 in Lemma 7.2 and Statement 8 in Lemma 7.5, $\{A_1, A_2\}$ is good for p. □

Lemma 8.3 *Suppose that $\{u_1, u_2\} \notin E(C_i)$ and $|E(C_i)| \geq 7$. Then, we can easily compute a maximal available matching-pair $\{A_1, A_2\}$ at time i that satisfies Conditions (C1) and (C2) and is good for p. Moreover, $\{A_1, A_2\}$ is good for every dangerous pair p' such that $Q(p')$ is of length 1 and $\tilde{Q}(p)$ and $\tilde{Q}(p')$ are vertex-disjoint.*

Proof. Let p' be as described in the lemma. We distinguish two cases as follows.

Case 1: Some edge e of $Q(p)$ incident to an endpoint of $Q(p)$ is available at time i. Let $e_1, \ldots, e_t$ be an ordering of the edges in C_i (appearing in C_i in this order) such that the edges in

$Q(p)$ are e_2, e_3, e_4 and $e_4 = e$. By definition, $\{e_1, e_5\}$ is available. Let (Z_1, Z_2) be the output of $FindMatch(i, \emptyset, \emptyset, C_i, e_1)$ where the ordering $e_1, \ldots, e_t$ is used at Step 1 in Figure 4. Let $\{A_1, A_2\}$ be a maximal available matching-pair at time i with $Z_1 \subseteq A_1$ and $Z_2 \subseteq A_2$. Then, by the first four statements in Lemma 7.5, $\{A_1, A_2\}$ satisfies Conditions (C1) and (C2). Moreover, by Lemma 7.2 and Statements 9 and 10 in Lemma 7.5, $\{A_1, A_2\}$ is good for p. Furthermore, by Lemma 7.2 and Statement 8 in Lemma 7.5, $\{A_1, A_2\}$ is good for p'.

Case 2: No edge of $Q(p)$ incident to an endpoint of $Q(p)$ is available at time i. Let $e_1, \ldots, e_5$ be an ordering of the edges in the extended witness path of p (appearing in C_i in this order) such that $E(Q(p)) = \{e_2, e_3, e_4\}$. Then, neither $\{e_2\}$ nor $\{e_4\}$ is available. By Lemma 5.3, both $\{e_1, e_3\}$ and $\{e_3, e_5\}$ are available at time i. So, to compute $\{A_1, A_2\}$, we can proceed as follows: Initialize $Y_1 = \{e_1, e_3\}$ and $Y_2 = \{e_5\}$; Remove e_1 from Y_1 and let (A_1, A_2) be the output of $FindMatch(i, Y_1, Y_2, P, e_1)$, where P is the unique path in $C_i - \{e_2, \ldots, e_5\}$ with $|E(P)| \geq 1$. Then, $\{e_1, e_3\} \subseteq A_1$ by Statements 5 and 6 in Lemma 7.5. In turn, $\{A_1, A_2\}$ is good for p by Lemma 7.2. Moreover, by the first four statements in Lemma 7.5, $\{A_1, A_2\}$ satisfies Conditions (C1) and (C2). Furthermore, by Lemma 7.2 and Statement 7 in Lemma 7.5, $\{A_1, A_2\}$ is good for p'. □

Lemma 8.4 *Suppose that $\{u_1, u_2\} \notin E(C_i)$ and $|E(C_i)| = 6$. Then, we can easily compute a maximal available matching-pair $\{A_1, A_2\}$ at time i that satisfies Conditions (C1) and (C2) and is good for p.*

Proof. Let $e_1, \ldots, e_6$ be the edges in C_i (appearing in C_i in this order), where e_1 and e_2 are incident to u_1, and e_4 and e_5 are incident to u_2. Then, by definition, $\{e_1, e_5\}$ or $\{e_2, e_4\}$ is available at time i. If both $\{e_1, e_5\}$ and $\{e_2, e_4\}$ are available at time i, then we are done by first setting $A_1 = \{e_1, e_5\}$ and $A_2 = \{e_2, e_4\}$ and further extending them to maximal available sets at time i. So, assume that either $\{e_1, e_5\}$ or $\{e_2, e_4\}$ is available at time i. We assume that $\{e_1, e_5\}$ is available at time i but $\{e_2, e_4\}$ is not; the other case is symmetric. We distinguish two cases as follows:

Case 1: Neither $\{e_2\}$ nor $\{e_4\}$ is available at time i. Then, by Lemma 5.3, $\{e_1, e_3\}$ is available at time i. So, we are done by first setting $A_1 = \{e_1, e_3\}$ and $A_2 = \{e_5\}$ and further extending them to maximal available sets at time i.

Case 2: e_2 or e_4 is available at time i. We assume that e_4 is available at time i; the other case is symmetric. Let (A_1, A_2) be the output of $FindMatch(i, \emptyset, \emptyset, C_i, e_1)$ where the ordering $e_1, e_2, \ldots, e_6$ is used at Step 1 in Figure 4. Then, by the first four statements in Lemma 7.5, $\{A_1, A_2\}$ satisfies Conditions (C1) and (C2). Moreover, $\{A_1, A_2\}$ is good for p, by Lemma 7.2, Statements 9 and 10 in Lemma 7.5, and the fact that $\{e_2, e_4\}$ is not available at time i. □

Now, based on Lemmas 8.2, 8.3, 8.4, and 7.1, our algorithm processes C_i as shown in Figure 6.

1. If $\{u_1, u_2\} \in E(C_i)$, then compute $\{A_1, A_2\}$ as described in Lemma 8.2; otherwise, compute $\{A_1, A_2\}$ as described in the proofs of Lemmas 8.3 and 8.4.

2. Perform Steps 3 through 5 in Figure 5 in turn.

Figure 6. Processing C_i when there is only one dangerous pair.

Lemma 8.5 *Let S be the set of C_i-settled edges. Suppose that there is exactly one dangerous pair at time i and our algorithm processes C_i as shown in Figure 6. Then, $\mathcal{E}[w(S \cap M'_7)] \geq w(S)/6$.*

Proof. Same as the proof of Lemma 8.1 □

8.2 The Case with Two or Three Dangerous Pairs

In this case, our algorithm colors no vertex of C_i *red* and processes C_i as shown in Figure 7:

> **1.** Select a dangerous pair $\{(u_1, v_1), (u_2, v_2)\}$ at time i uniformly at random.
>
> **2.** Perform the steps in Figure 6 in turn. (Comment: Lemmas 8.2 through 8.4 still hold even if there are two or more dangerous pairs at time i.)

Figure 7. Processing C_i when there are two or three dangerous pairs.

Lemma 8.6 *Let S be the set of C_i-settled edges. Suppose that there are two or three dangerous pairs at time i and our algorithm processes C_i as shown in Figure 7. Then, $\mathcal{E}[w(S \cap M'_7)] \geq 5w(S)/36$.*

PROOF. Let S_1 be the set of edges in S that are active at time i. Let S_2 be the set of edges in S_1 that are dangerous at time i. Consider an edge $e = (u, v)$ in S. By Invariant (I5), $\Pr[e \in S_1] \geq \frac{1}{2}$. So, it remains to show that $\Pr[e \in M'_7 \mid e \in S_1] \geq \frac{5}{18}$. By the proof of Lemma 8.1, $\Pr[e \in M'_7 \mid e \in S_1 - S_2] \geq \frac{1}{3} > \frac{5}{18}$. In turn, it remains to show that $\Pr[e \in M'_7 \mid e \in S_2] \geq \frac{5}{18}$.

Let b be the number of dangerous pairs at time i. Let D_1 be the event that $e \in S_2$ and the dangerous pair containing e is selected at Step 1 in Figure 7. Let D_2 be the event that $e \in S_2$ but the dangerous pair containing e is not selected at Step 1 in Figure 7. Note that D_1 occurs with probability $\frac{1}{b} \cdot \Pr[e \in S_2]$ and D_2 occurs with probability $(1 - \frac{1}{b}) \cdot \Pr[e \in S_2]$. By the proof of Lemma 8.1, $\Pr[e \in M'_7 \mid D_1] \geq \frac{1}{3}$. On the other hand, $\Pr[e \in M'_7 \mid D_2] \geq \frac{1}{4}$. Thus, $\Pr[e \in M'_7 \mid e \in S_2] \geq \frac{1}{b} \cdot \frac{1}{3} + (1 - \frac{1}{b}) \cdot \frac{1}{4} \geq \frac{5}{18}$. □

8.3 The Case with Four or More Dangerous Pairs

This case is more complicated than the previous cases. In this case, the length of C_i is at least 8 and hence we can prove the following important lemma:

Lemma 8.7 *Let $p = \{(u_1, v_1), (u_2, v_2)\}$ be a dangerous pair at time i. Suppose that a maximal available matching-pair $\{A_1, A_2\}$ at time i covers all vertices of $Q(p)$ and satisfies Condition (G3) for p. Then, $\{A_1, A_2\}$ satisfies Condition (G1) for p.*

PROOF. By Condition (G3) and renaming, we can assume that the two edges incident to u_1 in C_i belong to $A_1 \cup A_2$. Let e_1 and e_2 be the two edges incident to u_1 in C_i, where e_2 is also an edge in $Q(p)$. Let e_3 be the edge of $Q(p)$ incident to u_2, and let e_4 be the other edge incident to u_2 in C_i. Note that $e_2 = e_3$ when the length of $Q(p)$ is 1.

Since A_1 and A_2 are matchings, we can assume that $e_1 \in A_1$ and $e_2 \in A_2$ (again by renaming). A simple but crucial observation is that A_2 contains at least one edge in $E(C_i) - E(Q(p))$. This observation follows from the maximality of A_2 and Corollary 5.2 immediately. By this observation, the graph $C_i - A_2$ has no path between u_1 and u_2. If $E(Q(p)) \cap A_1 \neq \emptyset$, then $C_i - A_1$ has no path between u_1 and u_2, either. So, it remains to consider the case where $E(Q(p)) \cap A_1 = \emptyset$.

Suppose that $E(Q(p)) \cap A_1 = \emptyset$. Then, $e_3 \in A_2$ because A_2 is a matching and each vertex of $Q(p)$ is incident to an edge in $A_1 \cup A_2$. In turn, by Condition (G3), $e_4 \notin A_1 \cup A_2$. So, the degree of u_2 in the graph $C_i - A_1$ is 2. Thus, $\{A_1, A_2\}$ satisfies Condition (G1) for p. □

Suppose that C_i has been embedded in the plane. For each dangerous pair p at time i, the *left endpoint* of $Q(p)$ is the endpoint u of $Q(p)$ such that the other endpoint (called the *right endpoint*) of $Q(p)$ can be reached by starting at u, proceeding clockwise around C_i, and traversing the edges of $Q(p)$ only.

We construct a graph H_4 as follows. The nodes of H_4 are the dangerous pairs at time i. For two dangerous pairs p_1 and p_2 at time i, $\{p_1, p_2\}$ is an edge in H_4 if and only if one of the following two conditions holds true:

- $E(Q(p_1)) \cap E(Q(p_2)) \neq \emptyset$.
- $E(Q(p_1)) \cap E(Q(p_2)) = \emptyset$, and C_i contains a path P from some endpoint u of $Q(p_1)$ to some endpoint v of $Q(p_2)$ such that $E(P) \cap E(Q(p_1)) = E(P) \cap E(Q(p_2)) = \emptyset$ and P contains at most one other dangerous vertex than u and v.

A simple inspection shows that the degree of each node in H_4 is at most 5. Moreover, if p is a node of degree 5 in H_4, then p has a neighbor of degree 3 in H_4. Thus, the nodes of H_4 can be colored with at most five colors so that no two adjacent nodes get the same color.

Now, our algorithm processes C_i as follows:

1. Partition the node set of H_4 into at most five (nonempty) independent sets of H_4, and then select one independent set I among them uniformly at random.
2. Let H_5 be the graph $C_i - (\cup_{p \in I} V(Q(p)))$. (Comment: By the construction of H_4 and the independence of I, each connected component of H_5 contains at least two dangerous vertices at time i and hence is a path of length at least 1.)
3. Let J be those $p \in I$ with $|E(Q(p))| = 3$.
4. If $J = \emptyset$, then perform the following steps:
   (a) If H_5 has a connected component K with $|E(K)| \geq 2$, then perform the following steps:
      i. Find an available edge $e \in E(K)$ at time i. (Comment: By Lemma 5.1, e exists.)
      ii. Let $e_1, \ldots, e_t$ be the edges of C_i (appearing in C_i in this order), where $e_1 = e$ and $e_t \in E(K)$.
      iii. Let (A_1, A_2) be the output of $FindMatch(i, \emptyset, \emptyset, C_i, e)$.
      iv. Extend A_1 and A_2 to maximal available sets at time i. (Comment: By the first four statements in Lemma 7.5, $\{A_1, A_2\}$ satisfies Conditions (C1) and (C2). Moreover, by Lemma 7.2 and Statement 8 in Lemma 7.5, $\{A_1, A_2\}$ is good for all $p \in I$.)

   (b) If every connected component K of H_5 satisfies $|E(K)| = 1$, then perform the following steps:
      i. Choose an arbitrary dangerous pair $q = \{(u_1, v_1), (u_2, v_2)\} \notin I$ at time i. (Comment: Since every connected component of H_5 is an edge and $|E(C_i)| \geq 8$, $|I| \geq 2$. In turn, by the independence of I, q must exist.)
      ii. Choose one integer $j \in \{1, 2\}$ uniformly at random.
      iii. Color vertex u_j and edge (u_j, v_j) *red*.
      iv. Let e' be the edge of H_5 incident to u_j. Let e be the edge in $E(C_i) - \{e'\}$ adjacent to e' but not incident to u_j. (Comment: By the definition of a dangerous pair at time i, e is available at time i because $e \in E(\tilde{Q}(p)) - E(Q(p))$ for some $p \in I$.)
      v. Let (A_1, A_2) be the output of $FindMatch(i, \emptyset, \emptyset, C_i - \{e'\}, e)$, and extend A_1 and A_2 to maximal available sets at time i. (Comment: Note that one vertex of C_i is *red* now. By the first four statements in Lemma 7.5, $\{A_1, A_2\}$ satisfies Conditions (C1) and (C2). So, by Lemma 7.2 and Statement 7 in Lemma 7.5, $\{A_1, A_2\}$ is good for all $p \in I$.)

5. If J contains exactly one dangerous pair p, then compute A_1 and A_2 as described in Lemma 8.3. (Comment: Lemma 8.3 still holds for p, even if there are other dangerous pairs at time i. By this lemma, $\{A_1, A_2\}$ satisfies Conditions (C1) and (C2) and is good for every $p \in I$.)

6. If $|J| \geq 2$, then perform the following steps:

   (a) Let $p_1, \ldots, p_k$ be the dangerous pairs (nodes) in J (appearing in C_i in this order clockwise).

   (b) Let H_6 be the graph $C_i - (\cup_{1 \leq j \leq k} E(Q(p_i)))$. (Comment: By the construction of H_4 and the independence of I, each connected component K of H_6 with $|E(K)| = 0$ is a vertex of $Q(p_j)$ for some $j \in \{1, \ldots, k\}$, and each connected component K of H_6 with $|E(K)| > 0$ is a path of length at least 3.)

   (c) For each $j \in \{1, \ldots, k\}$, color the dangerous vertex at time i closest to the left endpoint of $Q(p_j)$ in H_6 *olive*, and color the dangerous vertex at time i closest to the right endpoint of $Q(p_j)$ in H_6 *brown*. (Comment: By the construction of H_4 and the independence of I, no vertex is colored twice at this step. For the same reason, for each vertex u colored at this step, the dangerous pair containing the dangerous edge incident to u is not in I. Moreover, there is no dangerous pair $\{(u_1, v_1), (u_2, v_2)\}$ such that u_1 and u_2 are assigned the same color at this step.)

   (d) Let U_O be the set of *olive* vertices. Let U_B be the set of *brown* vertices.

   (e) For each dangerous pair $p = \{(u_1, v_1), (u_2, v_2)\} \notin I$ at time i such that exactly one of u_1 and u_2 is colored, color the uncolored vertex in $\{u_1, u_2\}$ with a suitable color in $\{$*olive*,*brown*$\}$ so that u_1 and u_2 get different colors. (Comment: Immediately after this step, for each dangerous pair $p = \{(u_1, v_1), (u_2, v_2)\} \notin I$ at time i, either both u_1 and u_2 are uncolored, or they are colored with different colors.)

   (f) Select a color c among *olive* and *brown* uniformly at random.

   (g) If c is *olive*, then set $U_R = U_O$ and color all edges in M' incident to *olive* vertices *red*; otherwise, set $U_R = U_B$ and color all edges in M' incident to *brown* vertices *red*. (Comment: All *red* edges are dangerous at time i.)

   (h) Recolor the vertices in U_R *red*. (Comment: Some *red* edges may have no *red* endpoints.)

   (i) Compute an available matching-pair $\{A_1, A_2\}$ that satisfies Condition (C2) and is good for all $p \in I$.

   (j) Extend A_1 and A_2 to maximal available sets at time i. (Comment: By Corollary 5.2, $\{A_1, A_2\}$ satisfies Condition (C1) after this step. So, after this step, $\{A_1, A_2\}$ satisfies Conditions (C1) and (C2) and is good for all $p \in I$.)

7. Perform Steps 3 through 5 in Figure 5 in turn.

Step 6i is rough; the remainder of this subsection is devoted to it. Hereafter, we assume that $|J| \geq 2$. The following lemma is clear from our choice of *olive* vertices and *brown* vertices at Step 6c.

Lemma 8.8 *The following hold:*

1. *For every connected component (path) K of graph $C_i - U_R$, there is exactly one dangerous pair $p \in J$ with $E(Q(p)) \subseteq E(K)$.* (Comment: *Hereafter, we denote this dangerous pair by $p(K)$ for convenience.)*

2. *$C_i - U_R$ has no connected component K equal to $Q(p(K))$.*

3. *For each $p \in I - J$, $C_i - U_R$ has a connected component K with $E(Q(p)) \subset E(K)$, and neither endpoint of $Q(p)$ is an endpoint of path K.*

To compute A_1 and A_2 as required in Step 6i, our algorithm first initializes A_1 and A_2 to be empty, and then processes the connected components of $C_i - U_R$ one by one (in an arbitrary order). In a nutshell, during processing a connected component K of $C_i - U_R$, our algorithm does nothing else but the following two jobs:

(J1) Add some edges in $E(K) \cup \{f_1, f_2\}$ to A_1 and A_2, where f_1 and f_2 are the two edges in $E(C_i) - E(K)$ incident to an endpoint of path K.

(J2) Delete zero or more edges in $\{f'_1, f'_2\}$ from A_1 or A_2, or move zero or more suitable edges in $\{f'_1, f'_2\}$ from one of A_1 and A_2 to the other, where f'_1 (respectively, f'_2) is the edge in $E(C_i) - E(K)$ adjacent to f_1 (respectively, f_2).

The addition in Job (J1) is required to satisfy the following two conditions:

(C4) Immediately after the addition, $\{A_1, A_2\}$ is an available matching-pair at time i with $A_1 \cap A_2 = \emptyset$, covers all vertices of K, and is not harmful for all $p \in I$ with $E(Q(p)) \subseteq E(K)$.

(C5) Even if one or both of the two edges in $E(C_i) - E(K)$ incident to an endpoint of path K are deleted from A_1 or A_2, or are moved from one of A_1 and A_2 to the other in the future, $\{A_1, A_2\}$ will remain to be not harmful for all $p \in I$ with $E(Q(p)) \subseteq E(K)$.

The deletion in Job (J2) is required to satisfy the following condition:

(C6) If a non-*red* vertex of C_i was covered by $\{A_1, A_2\}$ before the deletion, then it remains to be covered by $\{A_1, A_2\}$ after the deletion.)

Lemma 8.9 *After processing all connected components K of $C_i - U_R$ as above, $\{A_1, A_2\}$ is an available matching-pair at time i, satisfies Condition (C2), and is good for all $p \in I$.*

Proof. Immediate from Conditions (C4) and (C6) and Lemma 7.2. □

We next detail how our algorithm does Jobs (J1) and (J2). The idea is similar to that in the proof of Lemma 8.3. To do Jobs (J1) and (J2) for a connected component K of $C_i - U_R$, our algorithm performs the following steps:

8. If there are distinct edges e_1 and e_2 in $Q(p(K))$ such that both $A_1 \cup \{e_1\}$ and $A_2 \cup \{e_2\}$ are available at time i, then perform the following steps:
   (a) For each $j \in \{1, 2\}$, add e_j to A_j, and find the path P_j in C_i such that $e_j \in E(P_j)$, $e_{j'} \notin E(P_j)$, and $|E(P_j) - E(K)| = 1$, where j' is the integer in $\{1, 2\} - \{j\}$.
   (b) For each $j \in \{1, 2\}$ (in any order), remove e_j from A_j, call $FindMatch(i, A_j, A_{j'}, P_j, e_j)$ to obtain (Z_1, Z_2), add the edges in Z_1 to A_j, and add the edges in Z_2 to $A_{j'}$, where j' is the integer in $\{1, 2\} - \{j\}$.
   (Comment: By procedure *FindMatch*, it is easy to see that what our algorithm does at Steps 8a and 8b are exactly Jobs (J1) and (J2). Moreover, immediately after Step 8b, $e_1 \in A_1$ and $e_2 \in A_2$ (implying that $\{A_1, A_2\}$ is not harmful for $p(K)$) by Statement 5 in Lemma 7.5, $\{A_1, A_2\}$ covers all vertices of K by Statement 2 in Lemma 7.5, and $\{A_1, A_2\}$ is not harmful for all $p \in I - \{p(K)\}$ with $E(Q(p)) \subseteq E(K)$ by Statement 3 in Lemma 8.8 and Statement 7 in Lemma 7.5. So, by Statement 1 in Lemma 7.5,

$\{A_1, A_2\}$ satisfies Condition (C4). By the contents of Jobs (J1) and (J2) to be done for other connected components of $C_i - U_R$, edges e_1 and e_2 will remain in A_1 and A_2 forever, respectively. In turn, $\{A_1, A_2\}$ will remain to be not harmful for $p(K)$ forever. Furthermore, for each $q \in I - \{p(K)\}$ with $E(Q(q)) \subseteq E(K)$, because of Statement 3 in Lemma 8.8, the contents of Jobs (J1) and (J2) to be done for other connected components of $C_i - U_R$ guarantee that $\{A_1, A_2\}$ will remain to be not harmful for q forever. Thus, $\{A_1, A_2\}$ satisfies Condition (C5). Finally, by Statement 6 in Lemma 7.5, $\{A_1, A_2\}$ satisfies Condition (C6).)

9. If there are no distinct edges e_1 and e_2 in $Q(p(K))$ such that both $A_1 \cup \{e_1\}$ and $A_2 \cup \{e_2\}$ are available at time i, then perform the following steps:

   (a) Let $e_1, \ldots, e_5$ be the edges of $\tilde{Q}(p(K))$ (appearing in C_i in this order), where $Q(p(K)) = \{e_2, e_3, e_4\}$ and $e_5 \in E(K)$.
   (Comment: By Statement 2 in Lemma 8.8, e_5 exists. By Lemma 5.1, both $A_1 \cup \{e_3\}$ and $A_2 \cup \{e_3\}$ are available at time i, and neither $A_j \cup \{e_2\}$ nor $A_j \cup \{e_4\}$ is available at time i for each $j \in \{1, 2\}$. So, by Lemma 5.3, $A_j \cup \{e_3, e_5\}$ is available at time i for each $j \in \{1, 2\}$ such that A_j does not contain the edge in $E(C_i) - \{e_1, e_2\}$ adjacent to e_1.)

   (b) Find an integer $j \in \{1, 2\}$ such that $A_j \cup \{e_3, e_5\}$ is available at time i. (Comment: By the comment on Step 9a and the fact that $A_1 \cap A_2 = \emptyset$, j exists.)

   (c) Add both e_3 and e_5 to A_j.

   (d) Let P_1 be the path in C_i such that $\{e_3, e_4\} \cap E(P_1) = \{e_3\}$ and $|E(P_1) - E(K)| = 1$. Let P_2 be the path in C_i such that $\{e_4, e_5\} \cap E(P_2) = \{e_5\}$ and $|E(P_2) - E(K)| = 1$. Let j' be the integer in $\{1, 2\} - \{j\}$.

   (e) Remove e_3 from A_j, call $FindMatch(i, A_j, A_{j'}, P_1, e_3)$ to obtain (Z_1, Z_2), add the edges in Z_1 to A_j, and add the edges in Z_2 to $A_{j'}$.

   (f) Remove e_5 from A_j, call $FindMatch(i, A_j, A_{j'}, P_2, e_5)$ to obtain (Z_1, Z_2), add the edges in Z_1 to A_j, and add the edges in Z_2 to $A_{j'}$.
   (Comment: By procedure $FindMatch$, it is easy to see that what our algorithm does at Steps 9a and 9f are exactly Jobs (J1) and (J2). Moreover, immediately after Step 9f, $\{e_3, e_5\} \subseteq A_j$ (implying that $\{A_1, A_2\}$ is not harmful for $p(K)$) by Statement 5 in Lemma 7.5, $\{A_1, A_2\}$ covers all vertices of K by Statement 2 in Lemma 7.5, and $\{A_1, A_2\}$ is not harmful for all $p \in I - \{p(K)\}$ with $E(Q(p)) \subseteq E(K)$ by Statement 3 in Lemma 8.8 and Statement 7 in Lemma 7.5. So, by Statement 1 in Lemma 7.5, $\{A_1, A_2\}$ satisfies Condition (C4). By the contents of Jobs (J1) and (J2) to be done for other connected components of $C_i - U_R$, edges e_3 and e_5 will remain in A_j forever. In turn, $\{A_1, A_2\}$ will remain to be not harmful for $p(K)$ forever. Furthermore, for each $q \in I - \{p(K)\}$ with $E(Q(q)) \subseteq E(K)$, because of Statement 3 in Lemma 8.8, the contents of Jobs (J1) and (J2) to be done for other connected components of $C_i - U_R$ guarantee that $\{A_1, A_2\}$ will remain to be not harmful for q forever. Thus, $\{A_1, A_2\}$ satisfies Condition (C5). Finally, by Statement 6 in Lemma 7.5, $\{A_1, A_2\}$ satisfies Condition (C6).)

Lemma 8.10 *Let S be the set of C_i-settled edges. Suppose that there are four or more dangerous pairs at time i and our algorithm processes C_i as described above in this subsection. Then, $\mathcal{E}[w(S \cap M'_7)] \geq 11w(S)/80$.*

Proof. Let S_1 be the set of edges in S that are active at time i. Let S_2 be the set of edges in S_1 that are dangerous at time i. Consider an edge $e = (u_1, v_1)$ in S. By Invariant (I5),

$\Pr[e \in S_1] \geq \frac{1}{2}$. So, it remains to show that $\Pr[e \in M'_7 \mid e \in S_1] \geq \frac{11}{40}$. By the proof of Lemma 8.1, $\Pr[e \in M'_7 \mid e \in S_1 - S_2] \geq \frac{1}{3} > \frac{5}{18}$. In turn, it remains to show that $\Pr[e \in M'_7 \mid e \in S_2] \geq \frac{11}{40}$.

Assume that the event $e \in S_2$ has occurred. Let $p = \{(u_1, v_1), (u_2, v_2)\}$ be the dangerous pair at time i containing e. Let b be the number of independent sets into which the node set of H_4 has been partitioned at Step 1. Clearly, $\Pr[p \in I \mid e \in S_2] = \frac{1}{b}$ and $\Pr[p \notin I \mid e \in S_2] = \frac{b-1}{b}$. Consider three cases as follows:

Case 1: $p \in I$. By Steps 4 through 6 and Lemma 8.7, $\{A_1, A_2\}$ is strongly good for p. Thus, as in the proof of Lemma 8.1, we can show that given the event $p \in I$, the event $e \in M'_7$ occurs with probability at least $\frac{3}{8}$. That is, $\Pr[e \in M'_7 \mid p \in I] \geq \frac{3}{8}$.

Case 2: $p \notin I$ and $J = \emptyset$. Clearly, p is the dangerous pair q chosen at Step 4(b)i or not.

Case 2.1: $p \neq q$. Then, $\Pr[e \in M'_6 \mid p \notin I] \geq \frac{1}{2}$ and hence $\Pr[e \in M'_7 \mid p \notin I] \geq \frac{1}{4}$.

Case 2.2: $p = q$. Let D be the event that u_1 is not colored *red* at Step 4(b)iii. D occurs with probability $\frac{1}{2}$, and hence $\Pr[e \in M'_6 \mid p \notin I] \geq \frac{1}{4}$. Moreover, if D occurs, then edge (u_2, v_2) is colored *red* at Step 4(b)iii. Thus, given the event $e \in M'_6$, the event $(u_2, v_2) \in M'_6$ cannot occur and so $e \in M'_7$ with probability 1 (because no cycle in $\mathcal{C}$ can contain e immediately after Step 6 in Figure 2). Therefore, $\Pr[e \in M'_7 \mid p \notin I] \geq \frac{1}{4}$.

Case 3: $p \notin I$ and $|J| \geq 2$. Clearly, $u_1 \in U_O \cup U_B$ or not (cf. Step 6c).

Case 3.1: $u_1 \notin U_O \cup U_B$. Then, the event $u_1 \in U_R$ cannot occur. So, $\Pr[e \in M'_6 \mid p \notin I] \geq \frac{1}{2}$ and hence $\Pr[e \in M'_7 \mid p \notin I] \geq \frac{1}{4}$.

Case 3.2: $u_1 \in U_O \cup U_B$. Then, the event $u_1 \notin U_R$ occurs with probability $\frac{1}{2}$ and hence $\Pr[e \in M'_6 \mid p \notin I] \geq \frac{1}{4}$. A crucial point is that if $u_1 \notin U_R$, then edge (u_2, v_2) is *red* because of Steps 6e and 6g. So, given the event $u_1 \notin U_R$ and the event $e \in M'_6$, the event $e \in M'_7$ occurs with probability 1 (because after Step 6 in Figure 2, edge (u_2, v_2) is not in $\mathcal{C}$ and hence no cycle in $\mathcal{C}$ can contain e). Thus, $\Pr[e \in M'_7 \mid p \notin I] \geq \frac{1}{4}$.

By Cases 2.1, 2.2, 3.1, and 3.2, we always have $\Pr[e \in M'_7 \mid p \notin I] \geq \frac{1}{4}$. Combining this with the result in Case 1, we have $\Pr[e \in M'_7 \mid e \in S_2] \geq \frac{3}{8} \cdot \frac{1}{b} + \frac{1}{4} \cdot \frac{b-1}{b}$. Since $b \leq 5$, we now have $\Pr[e \in M'_7 \mid e \in S_2] \geq \frac{11}{40}$. □

9 The Result

Recall T, T_{int}, T_{ext}, and α (they are defined in the beginning of Section 3).

Lemma 9.1 *Let $\delta w(T)$ be the expected total weight of edges moved from $\mathcal{C}$ to M at Step 4 or 5 in Figure 2. Then, $\mathcal{E}[w(T_2)] \geq (0.5 + \delta)w(T)$ and $\mathcal{E}[w(T_3)] \geq ((1 - \delta) + \frac{11}{160}(1 - \alpha))w(T)$.*

Proof. Since T_2 contains M_c (a maximum-weight perfect matching of G) as a subset and also contains the edges moved from $\mathcal{C}$ at Step 4 or 5 in Figure 2, it is clear that $\mathcal{E}[w(T_2)] \geq (0.5+\delta)w(T)$.

Obviously, each edge in M' is either 4-cycle-closed or C_i-settled for some $i \in \{\ell + 1, \ldots, r\}$. By Lemmas 6.17, 8.1, 8.5, 8.6, and 8.10, we have $\mathcal{E}[w(M'_7)] \geq \frac{11}{80}w(M')$. Thus, after Step 7 in Figure 2, $\mathcal{E}[w(\mathcal{C})] \geq (1 - \delta)w(T) + \frac{11}{80}w(M')$. Now, since $w(M') \geq \frac{1}{2}w(T_{\text{ext}}) = \frac{1}{2}(1 - \alpha)w(T)$, we have $\mathcal{E}[w(T_3)] \geq ((1 - \delta) + \frac{11}{160}(1 - \alpha))w(T)$. □

Now, combining Fact 3.1 and Lemma 9.1 and noting that the running time of the algorithm is dominated by the $O(n^3)$-time needed for computing a maximum-weight cycle cover and two maximum-weight matchings, we have:

Theorem 9.2 *For any fixed $\epsilon > 0$, there is an $O(n^3)$-time approximation algorithm for Max TSP achieving an expected approximation ratio of $\frac{251(1-\epsilon)}{331-320\epsilon}$.*

References

[1] A. I. Barvinok, D. S. Johnson, G. J. Woeginger, and R. Woodroofe. Finding Maximum Length Tours under Polyhedral Norms. *Proceedings of the Sixth International Conference on Integer Programming and Combinatorial Optimization* (IPCO), Lecture Notes in Computer Science, **1412** (1998) 195–201.

[2] P. Chalasani and R. Motwani. Approximating Capacitated Routing and Delivery Problems. *SIAM Journal on Computing*, **28** (1999) 2133–2149.

[3] R. Hassin and S. Rubinstein. An Approximation Algorithm for the Maximum Traveling Salesman Problem. *Information Processing Letters*, **67** (1998) 125–130.

[4] R. Hassin and S. Rubinstein. Better Approximations for Max TSP. *Information Processing Letters*, **75** (2000) 181–186.

[5] R. Hassin and S. Rubinstein. A 7/8-Approximation Approximations for Metric Max TSP. *Information Processing Letters*, **81** (2002) 247–251.

[6] A. V. Kostochka and A. I. Serdyukov. Polynomial Algorithms with the Estimates $\frac{3}{4}$ and $\frac{5}{6}$ for the Traveling Salesman Problem of Maximum (in Russian). *Upravlyaemye Sistemy*, **26** (1985) 55–59.

[7] A. I. Serdyukov. An Algorithm with an Estimate for the Traveling Salesman Problem of Maximum (in Russian). *Upravlyaemye Sistemy*, **25** (1984) 80–86.