An Approximation Algorithm for the Exact Matching Problem in Bipartite Graphs

Anita Dürr
Department of Computer Science, ETH Zürich, Switzerland

Nicolas El Maalouly
Department of Computer Science, ETH Zürich, Switzerland

Lasse Wulf
Institute of Discrete Mathematics, TU Graz, Austria

Abstract

In 1982 Papadimitriou and Yannakakis introduced the Exact Matching problem, in which given a red and blue edge-colored graph G and an integer k one has to decide whether there exists a perfect matching in G with exactly k red edges. Even though a randomized polynomial-time algorithm for this problem was quickly found a few years later, it is still unknown today whether a deterministic polynomial-time algorithm exists. This makes the Exact Matching problem an important candidate to test the **RP**=**P** hypothesis.

In this paper we focus on approximating Exact Matching. While there exists a simple algorithm that computes in deterministic polynomial-time an *almost* perfect matching with exactly k red edges, not a lot of work focuses on computing perfect matchings with *almost* k red edges. In fact such an algorithm for bipartite graphs running in deterministic polynomial-time was published only recently (STACS'23). It outputs a perfect matching with k' red edges with the guarantee that $0.5k \leq k' \leq 1.5k$. In the present paper we aim at approximating the number of red edges without exceeding the limit of k red edges. We construct a deterministic polynomial-time algorithm, which on bipartite graphs computes a perfect matching with k' red edges such that $\frac{k}{3} \leq k' \leq k$.

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

Keywords and phrases Perfect Matching, Exact Matching, Red-Blue Matching, Approximation Algorithms, Bounded Color Matching

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.18

Category APPROX

Related Version *Full Version*: http://arxiv.org/abs/2307.02205

Funding *Lasse Wulf*: Supported by the Austrian Science Fund (FWF): W1230.

1 Introduction

In the Exact Matching problem (denoted as EM) one is given a graph G whose edges are colored in red or blue and an integer k and has to decide whether there exists a perfect matching M in G containing exactly k red edges. This problem was first introduced in 1982 by Papadimitriou and Yannakakis [20] who conjectured it to be **NP**-complete. However a few years later, Mulmuley et al. [19] showed that the problem can be solved in randomized polynomial-time, making it a member of the class **RP**. This makes it unlikely to be **NP**-hard. In fact, while it is commonly believed that **RP**=**P**, the proof of this statement is a major open problem in complexity theory. Since EM is contained in **RP** but a deterministic polynomial-time algorithm for it is not known, the problem is a good candidate for testing the **RP**=**P** hypothesis. Actually Mulmuley et al. [19] even showed that EM is contained in **RNC**, which is the class of decision problems that can be solved by a polylogarithmic-time

algorithm using polynomially many parallel processors while having access to randomness. As studied in [22], derandomizing the perfect matching problem from this complexity class is also a big open problem. This makes EM even more intriguing as randomness allows it to be efficiently parallelizable but we don't even know how to solve it sequentially without randomness.

Due to this, numerous works cite EM as an open problem. This includes the above mentioned seminal work on the parallel computation complexity of the matching problem [22], planarizing gadgets for perfect matchings [14], multicriteria optimization [13], matroid intersection [5], DNA sequencing [4], binary linear equation systems with small hamming weight [2], recoverable robust assignment [10] as well as more general constrained matching problems [3, 17, 21, 18]. Progress in finding deterministic algorithms for EM has only been made for very restricted classes of graphs, thus illustrating the difficulty of the problem. There exists a deterministic polynomial-time algorithm for EM in planar graphs and more generally in $K_{3,3}$-minor free graphs [25], as well as for graphs of bounded genus [11]. Additionally, standard dynamic programming techniques on the tree decomposition of the graph can be used to solve EM on graphs of bounded treewidth [7, 23]. Contrary to those classes of sparse graphs, EM on dense graphs seems to be harder to solve. At least 4 articles study solely complete and complete bipartite graphs [16, 12, 24, 15]. More recently those results were generalized in [8] with a polynomial-time algorithm solving EM for constant independence number[1] α and constant bipartite independence number[2] β. These algorithm have **XP**-time, i.e. they run in time $O(g(\alpha)n^{f(\alpha)})$ and $O(g(\beta)n^{f(\beta)})$. This was further improved for bipartite graphs in [9] where the authors showed an FPT algorithm parameterized by the bipartite independence number β solving EM on bipartite graphs.

Approximating EM

Given the unknown complexity status of EM, it is natural to try approximating it. In order to approximate EM, we need to allow for non-optimal solutions. Here two directions can be taken: either we consider non-perfect matchings, or we consider perfect matchings with not exactly k red edges. Yuster [25] took the first approach and showed an algorithm that computes an *almost perfect exact matching* (a matching with exactly k red edges and either $\frac{n}{2}$ or $\frac{n}{2}-1$ total edges[3]). This is as close as possible for this type of approximation. However, there are two things to consider with Yuster's algorithm: First, the algorithm itself is very simple. Secondly, as long as a trivial condition on k is met, such an almost perfect exact matching always exists. It is surprising that this approximation is achieved by such simple methods in such generality. For the closely related *budgeted matching* problem, more sophisticated methods have been used recently to achieve a PTAS [3] and efficient PTAS [1]. These methods also do not guarantee to return a perfect matching.

The alternative is to find approximations which always return a perfect matching, but with possibly the wrong number of red edges. These kinds of approximations seem considerably more difficult to tackle. It is puzzling how little is known about this type of approximation, while the other type of approximation admits much stronger results. Very recently, El

[1] The independence number of a graph G is defined as the largest number α such that G contains an independent set of size α.

[2] The bipartite independence number of a bipartite graph G is defined as the largest number β such that G contains a balanced independent set of size 2β, i.e. an independent set using exactly β vertices from each partition.

[3] n is the number of vertices in the graph.

Maalouly [7] presented a polynomial-time algorithm which for bipartite graphs outputs a perfect matching containing k' red edges with $0.5k \leq k' \leq 1.5k$. To the best of our knowledge, this is the only result in this direction, despite the original EM problem being from 1982. Moreover, note that this algorithm can return both a perfect matching with too many or too few red edges. Hence El Maalouly's algorithm is not an approximation algorithm in the classical sense, as it only guarantees a two-sided, but not a one-sided, approximation.

Our contribution

We consider the problem of approximating EM, such that a perfect matching is always returned, and such that the approximation is one-sided. We show the first positive result of this kind. Formally, we consider the following optimization variant of EM:

Exact Matching Optimization (EM-opt)

Input: A graph G whose edges are colored red or blue; and an integer k.

Task: Maximize $|R(M)|$ subject to the constraint that M is a perfect matching in G and $|R(M)| \leq k$, where $R(M)$ is the set of red edges in M.

We consider bipartite graphs and prove:

▶ **Theorem 1.** *There exists a deterministic polynomial-time 3-approximation for EM-opt in bipartite graphs.*

We remark that there are three kinds of input instances to EM-opt : First, instances which are also YES-instances of EM, i.e. a perfect matching with k red edges exists. For those instances, our algorithm returns a perfect matching with k' red edges and $\frac{1}{3}k \leq k' \leq k$. Second, there are instances, where EM-opt is infeasible, in the sense that all perfect matchings have $k+1$ or more red edges. For such instances, our algorithm returns "infeasible". Finally, there are instances which are feasible, but the optimal solution to EM-opt has k^* red edges with $k^* < k$. For such instances, our algorithm returns a perfect matching with k' red edges and $\frac{1}{3}k^* \leq k' \leq k^*$.

The main idea behind our approximation algorithm is to start with an initial perfect matching, and then repeatedly try a small improvement step using dynamic programming. Whenever a small improvement step is not possible, we prove that the graph is "rigid" in a certain sense. We can prove that as a result of this rigidity, it is a good idea to guess a single edge e and to enforce that this edge e is contained in the solution matching. This paper contains concepts and lemmas, which make these ideas formal and may help for the development of future approximation algorithms. Our new ideas are based on a geometric intuition about the cycles appearing in such a "rigid" graph.

2 Preliminaries

2.1 Definitions and notations

All graphs considered are simple. Given a graph $G = (V, E)$, a perfect matching M of G is a subset of E such that every vertex in V is adjacent to exactly one edge in M. We introduce two tools related to M: the weight function w_M and the weighted directed graph G_M. These tools were first introduced in [8, 7]. They can be used to reason about EM, as is explained further below.

▶ **Definition** (Weight function w_M). *Given a graph $G = (V, E)$ whose edges are colored red or blue and a perfect matching M on G, we define the weight function w_M on the edges of G as follows:*

$$w_M(e) = \begin{cases} 0 & \text{if } e \text{ is a blue edge} \\ -1 & \text{if } e \in M \text{ is a red edge} \\ +1 & \text{if } e \notin M \text{ is a red edge} \end{cases}$$

▶ **Definition** (Oriented and edge-weighted graph G_M). *Given a bipartite graph $G = (A \sqcup B, E)$ whose edges are colored red or blue and a perfect matching M on G, the graph G_M is the oriented and edge-weighted graph such that:*

- *the vertex set of G_M is the vertex set $A \sqcup B$ of G;*
- *the edge set of G_M is the edge set of G such that edges in M are oriented from A to B and edges not in M are oriented from B to A;*
- *edges in G_M are weighted according to the weight function w_M.*

For ease of notation, we identify subgraphs of G and G_M with their edge sets. Any reference to their vertices will be made explicit. Since every edge in G_M has an undirected unweighted version in G, every subgraph H_M in G_M can be associated to the subgraph H in G of corresponding undirected and unweighted edge set, and vice versa. With a slight abuse of notation, we therefore sometimes do not differentiate H of H_M and denote them by the same object. If E_1 and E_2 are two edge sets, then we denote by $E_1 \Delta E_2 := (E_1 \setminus E_2) \cup (E_2 \setminus E_1)$ their symmetric difference. Let H be a subgraph of G_M. We use the following notations:

- $R(H)$ denotes the set of red edges in H.
- $E^+(H)$ denotes the set of positive edges in H (i.e. the red edges in H not contained in M).
- $E^-(H)$ denotes the set of negative edges in H (i.e. the red edges in H contained in M).
- $w_M(H)$ is the weight of H with respect to the edge weight function w_M, i.e. $w_M(H) = |E^+(H)| - |E^-(H)|$.

Unless stated otherwise, all considered cycles and paths in G_M are directed. Finally if C is a directed cycle in G_M, and $P_1 \subseteq C$ and $P_2 \subseteq C$ are sub-paths on C, we denote by $C[P_1, P_2]$ (respectively $C(P_1, P_2)$) the sub-path in C from P_1 to P_2 included (resp. excluded). If $w_M(C) > 0$ then we call C a *positive* cycle and if $w_M(C) \leq 0$ we call C a *non-positive* cycle. A *walk* in a directed graph is a sequence of edges $(e_1, \ldots, e_t)$ such that the end vertex of e_i is the start vertex of e_{i+1} for $i = 1, \ldots, t-1$. In contrast to a path, a walk may visit vertices and edges twice. A walk is *closed*, if its start and end vertex are equal.

2.2 Observations on G_M

In this subsection, we explain some simple observations, which we use later to reason about our approximation algorithm. Let G be a bipartite graph and M be a perfect matching of G. A cycle in G is said to be *M-alternating* if for any two adjacent edges in the cycle, one of them is in M and the other is not. It is a well-known fact that if M and M' are two perfect matchings, then $M \Delta M'$ is a set of vertex-disjoint cycles that are both M-alternating and M'-alternating. We now make the following important observations on the bipartite graphs G and G_M and the weight function w_M.

▶ **Observation 2.** *A cycle C in G is M-alternating if and only if the corresponding cycle C_M in G_M is directed.*

Proof. This follows directly from the definition of G_M. ◀

▶ **Observation 3.** *Let C be a directed cycle in G_M. Then $M' := M\Delta C$ is a perfect matching whose number of red edges is $|R(M')| = |R(M)| + w_M(C)$.*

Proof. Since C is M-alternating, it is a well-known fact that M' is again a perfect matching. The equation $|R(M')| = |R(M)| + w_M(C)$, follows directly from the definition of w_M. If an edge in C is blue, it does not change the amount of red edges of M. If an edge in C is red, it changes the number of red edges of M by ± 1, depending on whether or not it is in M. ◀

▶ **Observation 4.** *Let C_1 and C_2 be two directed cycles in G_M that intersect at a vertex v. Then C_1 and C_2 also intersect on an edge adjacent to v.*

Proof. By Observation 2, C_1 and C_2 are M-alternating so they both contain exactly one adjacent edge to v that is also contained in M. Since M is a perfect matching, there exists a single adjacent edge to v in M. Thus C_1 and C_2 both contain this edge. ◀

2.3 Previous ideas

The first part of our algorithm for Theorem 1 presented in Section 3 is strongly inspired by the algorithm in [7, Theorem 1] that computes a perfect matching with between $0.5k$ and $1.5k$ red edges. We summarize the main ideas here.

One can compute a perfect matching of minimum number of red edges in polynomial time by using a maximum weight perfect matching algorithm (e.g. [6]) on the graph G where red edges have weight 0 and blue edges have weight $+1$. The idea in [7] is to start with a perfect matching of minimum number of red edges M. If $0.5k \leq |R(M)| \leq 1.5k$ then we are already done and can output M. Otherwise $|R(M)| < 0.5k$ (it cannot have more than k red edges by minimality). In that case the algorithm iteratively improves M using positive cycles. Indeed, assume for a moment that we could find a directed cycle C in G_M such that both $w_M(C) > 0$ and $|E^+(C)| \leq k$, then the perfect matching $M' := M\Delta C$ is such that

$$\begin{aligned} |R(M)| < |R(M')| &= |R(M)| + w_M(C) \\ &\leq |R(M)| + |E^+(C)| < 0.5k + k = 1.5k. \end{aligned}$$

Thus we can iteratively compute such cycles C and replace M by $M\Delta C$. We make true progress every time until $0.5k \leq |R(M)| \leq 1.5k$.

An important observation proven in [7] is that we can determine in polynomial time if such a cycle exists. We recall this result in Proposition 5.

▶ **Proposition 5** (Adapted from [7, Proposition 11])**.** *Let $G := (V, E)$ be an edge-weighted directed graph and $t \in \mathbb{N}$ be a parameter. There exists a deterministic polynomial-time algorithm that, given G, determines whether or not there exists a directed cycle C in G with $w(C) > 0$ and $|E^+(C)| \leq t$. If such a cycle C exists then the algorithm also outputs a cycle C' with the same properties as C (i.e. $w(C') > 0$ and $|E^+(C')| \leq t$).*

The idea of Proposition 5 is to flip the sign of all weights so that we are looking for a negative cycle. Those can be found by shortest path algorithms: for example the Bellman-Ford algorithm which relies on a dynamic program (DP). Each entry of this DP contains the shortest distance between two vertices. The DP can be adapted so that it contains an additional budget constraint corresponding to the number of negative edges (i.e. positive before we flip the sign of the weight) a path is allowed to use. We also add in the entry of the DP the last edge used in the path. This way, when looking at the entries corresponding to the shortest path from a vertex v to the same vertex v we are able to determine whether

or not there exists a negative cycle that uses no more than t negative edges (i.e. positive before we flip the sign of the weight). If it is the case then the algorithm can output such a cycle. We redirect the reader for a detailed proof of Proposition 11 in [7, Section 3.1]. We finally remark that the proof in [7] is completed by showing that as long as M does not have between $0.5k$ and $1.5k$ red edges, a cycle C with $w_M(C) > 0$ and $|E^+(C)| \leq k$ always exists. However, to get a one-sided approximation, we need to guarantee the existence of a cycle C with $w_M(C) > 0$ and $|E^+(C)| \leq k - |R(M)|$. This might not be possible for certain graphs if $|R(M)| > 0$. This means that the above result is not enough for our purpose, as it only leads to a two-sided approximation.

2.4 Assumptions

In order to reduce the technical details needed to present our 3-approximation algorithm of EM-OPT, we show in this subsection that several simplifying assumptions can be made about the input instance.

▷ Claim 6. We can assume without loss of generality that the input instance to EM-OPT is also a YES-instance of EM, i.e. there exists a perfect matching with exactly k red edges.

Proof. Fix an instance $\langle G, k\rangle$ of EM-OPT. If the instance is feasible, let $k^* \leq k$ be the number of red edges in an optimal solution of EM-OPT. Suppose $\mathcal{A}$ is an algorithm for EM-OPT, which given an instance $\langle G, k\rangle$ returns a 3-approximate solution to EM-OPT if $k = k^*$, and returns some arbitrary perfect matching otherwise. We devise an algorithm $\mathcal{A}'$, which is a 3-approximation of EM-OPT for the input $\langle G, k\rangle$. First, $\mathcal{A}'$ computes in polynomial time the perfect matchings $M_{\min}$ and $M_{\max}$ with the minimum and maximum number of red edges. If it is not the case that $|R(M_{\min})| \leq k \leq |R(M_{\max})|$, then $\mathcal{A}'$ returns "infeasible". Afterwards, $\mathcal{A}'$ calls $\mathcal{A}$ as a subroutine for all $k' \in \{|R(M_{\min})|, \ldots, k\}$ and receives a perfect matching $M_{k'}$ each time. Finally, $\mathcal{A}'$ returns the best among all the matchings $M_{k'}$ that are feasible for EM-OPT. It is easy to see that $\mathcal{A}'$ is a correct 3-approximation, since there will be an iteration with $k' = k^*$. If $\mathcal{A}$ has running time $O(f)$, then $\mathcal{A}'$ has running time $O(nf + f_{\text{Mat}})$, where f_{Mat} denotes the time to deterministically compute a maximum weight perfect matching. ◁

▷ Claim 7. We can assume without loss of generality that for every edge $e \in E$ there exists a perfect matching of G containing e.

Proof. If an edge is not contained in any perfect matching, it is irrelevant and it can be deleted. We can therefore pre-process the graph and check for each edge in time $O(f'_{\text{Mat}})$ if it is irrelevant. Here f'_{Mat} denotes the time to deterministically check if a graph has a perfect matching, e.g. by running a maximum weight perfect matching algorithm. ◁

3 A 3-approximation of Exact Matching

In this section we are proving our main result, Theorem 1. Let $G = (A \sqcup B, E)$ be a red-blue edge-colored bipartite graph and k be an integer, such that they together form an instance of EM-OPT. We work under the assumptions of Section 2.4. In Algorithm 1 we show an algorithm that given G and k outputs a perfect matching containing between $\frac{1}{3}k$ and k red edges. Let us discuss the general ideas before formally analysing the algorithm.

We reuse the idea from [7] to iteratively improve a perfect matching M by a small amount until it has the right amount of red edges. This improvement is done by finding positive cycles in G_M with no more than $\frac{2}{3}k$ positive edges (using the algorithm of Proposition 5). As

Algorithm 1 The 3-approximation algorithm for EM-OPT of Theorem 1.

Input: A bipartite graph $G = (A \sqcup B, E)$ and an integer $k \geq 0$, such that they fulfill the assumptions from Section 2.4.

Output: M, a perfect matching such that $\frac{1}{3}k \leq |R(M)| \leq k$.

1 Compute a perfect matching M of minimal number of red edges.
2 If $|R(M)| \geq \frac{1}{3}k$ then output M.
3 Otherwise compute a cycle C in G_M with $w_M(C) > 0$ and $|E^+(C)| \leq \frac{2}{3}k$, or determine that no such cycle exists.
4 If such a cycle C exists, set $M \leftarrow M\Delta C$ and go to *Step 2*.
5 Otherwise there are no positive cycles with $|E^+(C)| \leq \frac{2}{3}k$ in G_M. For every red edge $e \in R(G)$ do the following:
6 Compute the perfect matching M^e containing e of minimal number of red edges.
7 If $\frac{1}{3}k \leq |R(M^e)| \leq k$ then output M^e.
8 Output $\perp$

opposed to the algorithm in [7], we do not want to "overshoot" the number of red edges, i.e. the obtained perfect matching cannot have more than k red edges. We therefore constrain the cycle to have $|E^+(C)| \leq \frac{2}{3}k$. This implies $|R(M\Delta C)| \leq k$. However the issue with that approach is that unlike in [7], we cannot guarantee that such a cycle always exists. So what do we do if we are not able find a good cycle? The answer to this question turns out to be the key insight behind our algorithm. For every edge e we define

$$M^e \in \arg\min\{|R(M)| : M \text{ is a perfect matching with } e \in M\}$$

to be a perfect matching of G containing e with minimal number of red edges. Note that by Claim 7, M^e is properly defined. We also note that M^e can be computed in polynomial time.

We claim that in the case that no good cycle is found, the set of matchings M^e "magically" fixes our problems. Specifically, we claim that at least one of the matchings M^e is a sufficient approximation. The rough intuition behind this is the following. The fact that no good cycle exists means that the graph does not contain small substructures, which can be used to modify the current solution M towards a better approximation. In a sense, the graph is rigid. This rigidity means that maybe there could exist some edge e in the optimal solution M^*, such that every perfect matching including e is already quite similar to M^*. Our approximation algorithm simply tries to guess this edge e. After proving Lemma 8, the remaining part of the paper is devoted to quantify and prove this structural statement.

▶ **Lemma 8.** *Algorithm 1 runs in polynomial time and either outputs $\perp$ or a correct 3-approximation of EM-OPT.*

Proof. Note that the first inner loop is executed at most $O(|V|)$ times, since $w_M(C) > 0$ in each iteration, thus $|R(M)|$ is monotonously increasing. Furthermore, every iteration of the loop is executed in polynomial time due to Proposition 5. The second loop is executed at most $O(|E|)$ times. The matching in *Step 1*, as well as the matchings M^e can be computed in polynomial time using a minimum weight perfect matching algorithm. In total, this takes polynomial time. If the algorithm does not output $\perp$, then it outputs either in *Step 7* (which is obviously a 3-approximation), or in *Step 2*, which is a 3-approximation because $k/3 \leq |R(M\Delta C)| = |R(M)| + w_M(C) \leq |R(M)| + |E^+(C)| \leq k$. ◀

All it remains to prove is that Algorithm 1 never outputs $\perp$. This happens if the conditions in *Step 2* and in *Step 7* are not satisfied. In that regard, define the following *critical tuple*.

▶ **Definition** (Critical tuple). *Let (G, k, M^*, M) be such that G is a bipartite graph whose edges are colored red or blue, $k \geq 0$ is an integer and M^* and M are perfect matchings of G. The tuple (G, k, M^*, M) is said to be* critical *if:*

- *All the assumptions in Section 2.4 hold.*
- *M^* is a perfect matching of G with exactly k red edges.*
- *$|R(M)| < \frac{1}{3}k$.*
- *If C is a directed cycle in G_M such that $w_M(C) > 0$ then $|E^+(C)| > \frac{2}{3}k$.*
- *For every edge $e \in R(M^* \setminus M)$ we have $|R(M^e)| < \frac{1}{3}k$.*

Note that if Algorithm 1 outputs $\perp$, we must have a critcal tuple. Indeed, for every edge $e \in R(M^* \setminus M)$, by minimality of M^e, we have $|R(M^e)| \leq k$. So if the algorithm returns $\perp$, then it must be the case that $|R(M^e)| < k/3$ for all $e \in R(G)$ so in particular also for all $e \in R(M^* \setminus M)$. We will prove the following: critical tuples do not exist. This means that Algorithm 1 never outputs $\perp$ and is therefore a correct 3-approximation algorithm.

▶ **Lemma 9.** *A tuple (G, k, M^*, M) can never be critical.*

3.1 Overview of the main proof

In this section, we explain the idea behind the proof of our main lemma, Lemma 9. From this point onward, consider a fixed tuple (G, k, M^*, M). We prove Lemma 9 by contradiction and therefore assume that (G, k, M^*, M) is a critical tuple. The main strategy of the proof is to find a set of cycles with contradictory properties. Such a set of cycles is called a *target set*. We first introduce a special cycle, which we call the cycle C^+.

▶ **Definition** (Cycle C^+). *C^+ is the positive cycle in G_M such that $C^+ \subseteq M \Delta M^*$.*

We claim that C^+ is well-defined and unique. Indeed, since $|R(M)| < |R(M^*)|$, the set of cycles $M \Delta M^*$ has to contain at least one positive cycle. Furthermore, there are at most $k = |R(M^*)|$ positive edges in $M \Delta M^*$ and by the definition of a critical tuple, every positive cycle contains at least $\frac{2}{3}k$ of them. Hence there is no second positive cycle in $M \Delta M^*$.

We list the following simple observations. The last two state that there only exist two different types of cycles in a critical tuple. These two types can be distinguished by examining the number of positive edges in a cycle C. We will repeatedly make use of this key observation.

▶ **Observation 10.** *Let (G, k, M^*, M) be a critical tuple. Then the following properties hold.*

1. *$|E^-(G_M)| < \frac{1}{3}k$.*
2. *$\frac{2}{3}k < |E^+(C^+)| \leq k$*
3. *If C is a directed cycle in G_M then $w_M(C) > 0$ if and only if $|E^+(C)| > \frac{2}{3}k$.*
4. *If C is a directed cycle in G_M then $w_M(C) \leq 0$ if and only if $|E^+(C)| < \frac{1}{3}k$.*

Proof. (G, k, M^*, M) is a critical tuple so in particular $|R(M)| < \frac{1}{3}k$, thus the graph G_M contains at most $\frac{1}{3}k$ negative edges. As a consequence, if C is a non-positive cycle then $|E^+(C)| \leq |E^-(C)|$ and thus $|E^+(C)| < \frac{1}{3}k$. Furthermore, in a critical tuple every positive cycle contains at least $\frac{2}{3}k$ positive edges, hence the third observation follows directly. In particular, since C^+ is a positive cycle, we have $\frac{2}{3}k < |E^+(C^+)|$. Finally the upper bound on $|E^+(C^+)|$ comes from the fact that positive edges in $M \Delta M^*$ are red edges in M^*, hence $M \Delta M^*$ contains at most $k = |R(M^*)|$ positive edges. In particualr $|E^+(C^+)| \leq k$. ◀

▶ **Definition** (Target set). *We call a set $\mathcal{C}$ of non-positive cycles in G_M a* target set *if:*

1. *$\forall e \in E^+(C^+)$ there exists a cycle in $\mathcal{C}$ containing e. (Condition 1)*
2. *$\forall C \in \mathcal{C}$, $C \cap C^+$ is a single path. (Condition 2)*
3. *$\forall e \in E^-(G_M)$ there are at most two cycles in $\mathcal{C}$ containing e. (Condition 3)*

The following Lemma 11 shows that in order to prove our main lemma, Lemma 9, it suffices to find a target set.

▶ **Lemma 11.** *Let (G, k, M^*, M) be a critical tuple and $\mathcal{C}$ be a target set in G_M. Then a contradiction arises.*

Proof. We can lower bound the sum of the weights of cycles in $\mathcal{C}$.

$$\begin{aligned}\sum_{C\in\mathcal{C}} w_M(C) &\geq 1\cdot|E^+(C^+)| - 2\cdot|E^-(G_M)|\\ &> \frac{2}{3}k - 2\cdot\frac{1}{3}k \qquad \text{(by Observation 10)}\\ &= 0\end{aligned}$$

However $\sum_{C\in\mathcal{C}} w_M(C) \leq 0$ since all cycles $C \in \mathcal{C}$ have non-positive weight $w_M(C) \leq 0$, so this is a contradiction. ◀

Note that Condition 2 is not needed in the proof of Lemma 11. However Condition 2 will be useful to achieve Condition 3.

The rest of the paper is structured as follows: First, we construct a set $\mathcal{C}$ of non-positive cycles satisfying only Condition 1. This is explained in Section 3.2. After that, we explain in Section 3.3 how to modify the initial set to additionally satisfy Condition 2. In Section 3.4 we show how to modify this set again to also satisfy Condition 3. Finally, the proof of Lemma 9 is summarized in Section 3.5. To help visualize the reasonings on G_M we provide various illustrations. We focus only on the orientation of paths and cycles and not on actual vertices and edges of G_M so we omit them and draw paths and cycles using simple lines whose orientation is indicated by an arrow. Individual edges and vertices will be indicated in red.

3.2 Obtaining an initial set of cycles satisfying Condition 1

We explain how to obtain an initial set of non-positive cycles in G_M which satisfies only Condition 1. To this end, for a fixed critical tuple (G, k, M^*, M) consider the following definition:

▶ **Definition** (Cycle C_e)**.** *Let $e \in E^+(C^+)$. We define $C_e \subseteq M\Delta M^e$ as the unique directed cycle in $M\Delta M^e$ which contains the edge e.*

Observe that C_e is well-defined, since $e \in M\Delta M^e$. We claim that for all possible $e \in E^+(C^+)$ we have $w_M(C_e) \leq 0$. Indeed, positive edges in $M\Delta M^e$ are red edges in M^e. By the definition of a critical tuple and $E^+(C^+) \subseteq R(M^* \setminus M)$, we have $|R(M^e)| < \frac{1}{3}k$, thus $|E^+(C_e)| < \frac{1}{3}k$. Hence by Observation 10, C_e has non-positive weight in G_M. As a direct consequence, the set of cycles $\{C_e : e \in E^+(C^+)\}$ is a set of non-positive cycles satisfying Condition 1.

3.3 Modifying the set to satisfy Condition 2

We now show how to modify the initial set $\{C_e : e \in E^+(C^+)\}$ into a set of non-positive cycles such that both Conditions 1 and 2 hold. Consider a fixed critical tuple (G, k, M^*, M). If every cycle C_e intersects C^+ in a single path, then we would directly have Condition 2. However, in reality the interaction between C_e and C^+ can be a great amount more complicated. One example is depicted in Figure 1. In order to analyse the interaction between these two

cycles, we define multiple notions around them. First, we decompose the cycle C_e into *jumps* and *interjumps* where jumps are the sub-paths of C_e outside of C^+ and interjumps are the sub-paths of C_e intersecting C^+ between jumps.

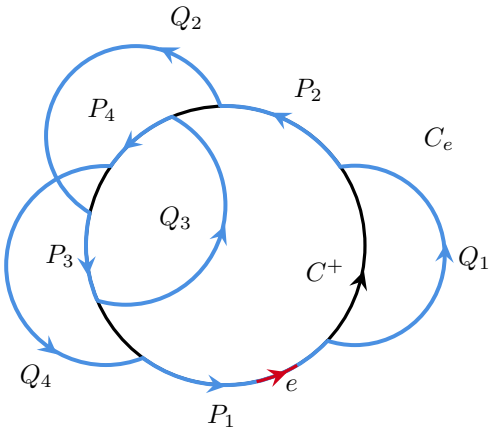


Figure 1 Decomposition of the cycle C_e (in blue) into 4 jumps Q_1, Q_2, Q_3, Q_4 and 4 interjumps P_1, P_2, P_3, P_4. The first interjump P_1 contains the edge e.

▶ **Definition** (Jumps and interjumps). *Let $e \in E^+(C^+)$. A jump of C_e is a sub-path $Q \subseteq C_e$ such that its endpoints are vertices of C^+ but none of its inner vertices are in C^+. Note that $Q \subseteq G_M \setminus C^+$. An interjump of C_e is a sub-path of $P \subseteq C_e$ contained in C^+ such that its endpoints are the endpoints of jumps of C_e. Note that $P \subseteq C^+$.*

▶ **Definition** (Decomposition of C_e into alternating jumps and interjumps). *Let $e \in E^+(C^+)$. Decompose C_e into jumps $Q_1, \ldots, Q_\ell$ and interjumps $P_1, \ldots P_\ell$ such that $e \in P_1$ and C_e is the concatenation of alternating jumps and interjumps:*

$$C_e = P_1 Q_1 \ldots P_j Q_j \ldots P_\ell Q_\ell$$

Let $\mathcal{J}_e = \{Q_1, Q_2, \ldots, Q_\ell\}$ be the set of jumps in C_e and $\mathcal{I}_e = \{P_1, P_2, \ldots, P_\ell\}$ be the set of interjumps in C_e.

Note that every jump in C_e must be followed by an interjump since all cycles considered are M-alternating cycles which cannot intersect in a single vertex (by Observation 4). Figure 1 shows a cycle C_e and its decomposition into jumps and interjumps.

To understand the interplay between C_e and C^+, we introduce the notion of forward and backward motion based on a visual intuition. Consider the directed cycle C^+ and take the convention of always drawing it with an anticlockwise orientation. We say that moving along C^+ in the direction of its directed edges equals to an *anticlockwise* or *forward* motion, while moving on the cycle C^+ against the direction of its directed edges equals to a *clockwise* or *backward* motion. We also want to interpret the direction a jump $Q \in \mathcal{J}_e$ is taking. However since Q is not a path in C^+ there is no obvious rule whether it should be seen as moving forward or backward. In fact, Figure 2 shows that the same jump can in principle be interpreted either as following a forward or a backward motion. We introduce the following rule which classifies all jumps as either a forward or a backward jump. We follow the convention to draw forward jumps outside of C^+, while we draw backward jumps inside of C^+. With this interpretation, in Figure 1 the jumps Q_1, Q_2 and Q_4 are forward and the jump Q_3 is backward.

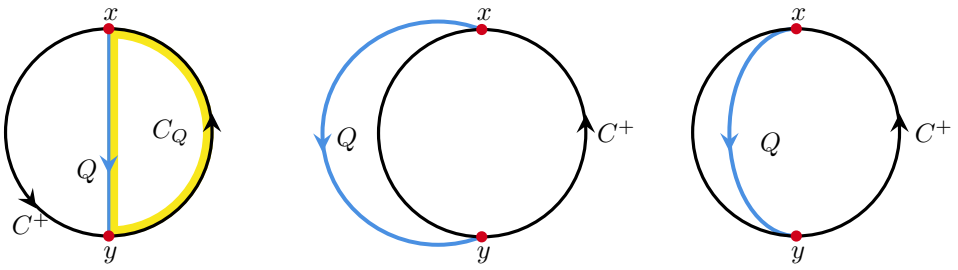


Figure 2 Consider a jump Q (blue) from the vertex x to the vertex y in C^+. Depending on the weight of C_Q (yellow highlight), the jump Q is either interpreted as following a forward (middle) or a backward (right) motion.

▶ **Definition** (Cycle C_Q and forward/backward jump). *Let $e \in E^+(C^+)$ and let Q be a jump of C_e. We define C_Q to be the unique directed cycle in the graph $C^+ \cup Q$ containing Q. Then:*

- *if $w_M(C_Q) > 0$, we call Q a* forward *jump;*
- *if $w_M(C_Q) \leq 0$, we call Q a* backward *jump.*

Let $\mathcal{J}_e^f$ be the set of forward jumps in C_e and $\mathcal{J}_e^b$ the set of backward jumps in C_e.

Observe that this is a valid definition, since the cycle C_Q is unique and independent of whether Q is a forward or a backward jump. We give a short informal explanation of why we use the weight of C_Q to distinguish between forward and backward jumps. A positive cycle must contain a large number of positive edges (at least $\frac{2}{3}k$ many, by Observation 10) and most of the positive edges of C_Q are contained in C^+ (since C_e has at most $\frac{1}{3}k$ positive edges). So for a forward jump Q_f, the cycle C_{Q_f} must have a large intersection with C^+ to be positive, i.e. Q_f only skips a small portion of the cycle. Similarly, for a backward jump Q_b, the cycle C_{Q_b} must be a non-positive cycle so it cannot contain too many positive edges and therefore cannot intersect a large part of C^+.

Finally, we introduce the notion of the *reach* of a jump Q, which is intuitively the sub-path of C^+ the jump Q is "jumping over". Figure 3 illustrates the reach of a forward jump and a backward jump. We say that the jump Q *covers* the edges in the reach of Q.

▶ **Definition** (Reach of a jump). *Let $e \in E^+(C^+)$ and let Q be a jump of C_e.*

- *If Q is a forward jump: the reach of Q is the sub-path of C^+ defined as $r(Q) := C^+ \setminus C_Q$.*
- *If Q is a backward jump: the reach of Q is the sub-path of C^+ defined as $r(Q) := C_Q \setminus Q$.*

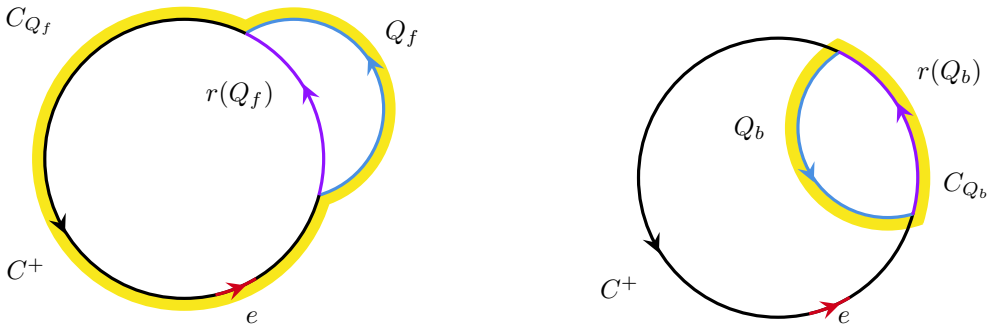


Figure 3 A forward jump Q_f and a backward jump Q_b (in blue). The respective reaches $r(Q_f)$ and $r(Q_b)$ of the jumps are colored in purple and the cycles C_{Q_f} and C_{Q_b} are highlighted in yellow.

We also extend the definition of the reach of a jump to the reach of C_e. Intuitively the reach of C_e is the union of sub-paths of C^+ the cycle C_e is jumping over. Here we differentiate between the sub-paths that are jumped over using *forward motions* (forward jumps or interjumps) and *backward motions* (backward jumps).

▶ **Definition** (Reach of C_e)**.** *Let $e \in E^+(C^+)$. The forward reach $r_f(C_e)$ of C_e is defined as the union of every interjump of C_e and the reach of every forward jump of C_e.*

$$r_f(C_e) := \bigcup_{Q \in \mathcal{J}_e^f} r(Q) \cup \bigcup_{P \in \mathcal{I}_e} P$$

The backward reach $r_b(C_e)$ of C_e is defined as the union of the reaches of every backward jump of C_e.

$$r_b(C_e) := \bigcup_{Q \in \mathcal{J}_e^b} r(Q)$$

The reach $r(C_e)$ of C_e is defined as the union of the forward reach and the backward reach of C_e.

$$r(C_e) := r_f(C_e) \cup r_b(C_e)$$

All above definitions still hold when replacing C_e by a sub-path $\mathcal{P}$ of C_e. Since the reach of a truncated jump is undefined, we will only consider sub-paths which endpoints are vertices of C^+. If $\mathcal{P}$ contains a sub-path of an interjump then the forward reach of $\mathcal{P}$ contains this sub-path and not the entire interjump.

With those newly defined notions, we note that to achieve Condition 2, it suffices to prove the following Lemma 12. Due to space contraints, we leave the proof to Appendix A.

▶ **Lemma 12.** *Let (G, k, M^*, M) be a critical tuple. Every edge $e \in E^+(C^+)$ is contained in the backward reach of C_e, i.e. $\forall e \in E^+(C^+) : e \in r_b(C_e)$.*

Indeed, we can then construct a set of non-positive cycles satisfying Conditions 1 and 2 as explained in Lemma 13.

▶ **Lemma 13.** *Let (G, k, M^*, M) be a critical tuple. There exists a set $\mathcal{C}$ of non-positive cycles in G_M satisfying Conditions 1 and 2, i.e. such that the following holds:*

- *$\forall e \in E^+(C^+)$ there exists a cycle in $\mathcal{C}$ containing e.*
- *$\forall C \in \mathcal{C}$, $C \cap C^+$ is a single path.*

Proof. By Lemma 12, for each edge $e \in E^+(C^+)$ there exists a backward jump Q_e in C_e that covers e. Since it is a backward jump, the cycle C_{Q_e} formed by Q_e and its reach is a non-positive cycle. It also contains e and intersects C^+ in a single path $r(Q_e)$. Hence the set $\mathcal{C} := \{C_{Q_e} : e \in E^+(C^+), Q_e \in \mathcal{J}_e^b, e \in r(Q_e)\}$ satisfies Conditions 1 and 2. ◀

3.4 Modifying the set to satisfy Condition 3

In the previous section we obtained a set $\mathcal{C}$ of non-positive cycles satisfying Conditions 1 and 2. To additionally satisfy Condition 3, which is that for every negative edge $e \in E^-(G_M)$ there exist at most two cycles in $\mathcal{C}$ containing e, we proceed in two steps. First we focus in Lemma 15 on bounding the number of cycles containing a negative edge on the cycle C^+. Then we prove in Lemma 16 how to bound the number of cycles containing a negative edge outside the cycle C^+.

Observation 14 shows with a simple argument that if a set of paths in C^+ covers the whole cycle C^+, then it is enough to keep at most two paths covering the same edge in C^+. This is the key idea behind Lemma 15.

▶ **Observation 14.** *If $\mathcal{P}'$ is a set of paths in C^+, then there exists a subset $\mathcal{P} \subseteq \mathcal{P}'$ such that $\bigcup_{P' \in \mathcal{P}'} P' = \bigcup_{P \in \mathcal{P}} P$ and every $e \in C^+$ is contained in at most two paths of $\mathcal{P}$.*

Proof. Let $X = \bigcup_{P' \in \mathcal{P}'} P'$. Take $\mathcal{P}$ to be a minimal subset of $\mathcal{P}'$ such that $\bigcup_{P \in \mathcal{P}} P = X$. Suppose there exists an edge $e \in C^+$ such that $\exists P_1, P_2, P_3 \in \mathcal{P}$ with $e \in P_1 \cap P_2 \cap P_3$. W.l.o.g. suppose P_1 contains an edge that is furthest before e on C^+ among all edges of P_1, P_2 and P_3, and P_2 contains an edge furthest after e. Then $P_3 \subseteq P_1 \cup P_2$, which contradicts the minimality of $\mathcal{P}$. So every edge is contained in at most two paths from $\mathcal{P}$. ◀

▶ **Lemma 15.** *Let (G, k, M^*, M) be a critical tuple and $\mathcal{C}$ be a set of non-positive cycles in G_M satisfying Conditions 1 and 2, i.e. such that:*

- *$\forall e \in E^+(C^+)$ there exists a cycle in $\mathcal{C}$ containing e.*
- *$\forall C \in \mathcal{C}$, $C \cap C^+$ is a single path.*

Then there exists a set $\mathcal{C}' \subseteq \mathcal{C}$ of non-positive cycles satisfying the above Conditions 1 and 2 as well as the following property:

- *$\forall e \in E^-(C^+)$ there are at most two cycles in $\mathcal{C}$ containing e.*

Proof. By Condition 2, we can associate to each cycle C of $\mathcal{C}$ the sub-path $P_C := C \cap C^+$ intersecting with C^+. Let $\mathcal{P} := \{P_C : C \in \mathcal{C}\}$ be the set of such intersecting paths. By Condition 1 for every edge $e \in E^+(C^+)$ there exists a path $P \in \mathcal{P}$ containing e. We can thus apply Observation 14 to $\mathcal{P}$ and get a set $\mathcal{P}' \subseteq \mathcal{P}$ such that for every edge $e \in E^+(C^+)$ there exists a path $P \in \mathcal{P}'$ containing e and every $e \in C^+$ is contained in at most two paths from $\mathcal{P}'$. Let $\mathcal{C}' := \{C : P_C \in \mathcal{P}'\}$ be the set of cycles associated to each path in $\mathcal{P}'$. Then $\mathcal{C}' \subseteq \mathcal{C}$ still satisfies Conditions 1 and 2 and additionally for every edge $e \in C^+$ at most two cycles in $\mathcal{C}'$ contain e. ◀

▶ **Lemma 16.** *Let (G, k, M^*, M) be a critical tuple and $\mathcal{C}$ be a set of non-positive cycles in G_M satisfying Conditions 1 and 2, i.e. such that:*

- *$\forall e \in E^+(C^+)$ there exists a cycle in $\mathcal{C}$ containing e.*
- *$\forall C \in \mathcal{C}$, $C \cap C^+$ is a single path.*

Then there exists a set $\mathcal{C}' \subseteq \mathcal{C}$ of non-positive cycles satisfying the above Conditions 1 and 2 as well as the following property:

- *$\forall e \in E^-(G_M \setminus C^+)$ there are at most two cycles in $\mathcal{C}$ containing e.*

Proof. We show how to reduce three non-positive cycles of $\mathcal{C}$ containing an edge $e \in E^-(G_M \setminus C^+)$ into two non-positive cycles such that, when replacing the three cycles by the two new cycles, Conditions 1 and 2 still hold. By repeating this transformation, the number of cycles containing a negative edge $e \in E^-(G_M \setminus C^+)$ decreases until eventually at most two cycles in $\mathcal{C}$ contain e.

Fix an edge $e \in E^-(G_M \setminus C^+)$ and three non-positive cycles C_{Q_1}, C_{Q_2} and C_{Q_3} in $\mathcal{C}$ all containing e. For $i \in \{1, 2, 3\}$, by Condition 2, C_{Q_i} intersects C^+ in a single path. Hence let $r(Q_i) = C_{Q_i} \cap C^+$ be this path and let $Q_i := C_{Q_i} \setminus C^+$ be the remaining path of C_{Q_i} not intersecting C^+. Let also s_i and f_i be the first and last edge of Q_i. We can assume w.l.o.g. that the first edge of $r(Q_i)$ appears anti-clockwise on C^+ in the order $1, 2, 3$.

For $(i, j) \in \{(1, 2), (2, 3), (3, 1)\}$ consider the walk $W_{ij} = Q_j[s_j, e] \cup Q_i(e, f_i]$ that goes from s_j to f_i. This is a valid walk since $e \in Q_1 \cap Q_2 \cap Q_3$. W_{ij} can use edges multiple times so we identify W_{ij} to its multi-edge set. As illustrated in Figure 4, we can extract a simple

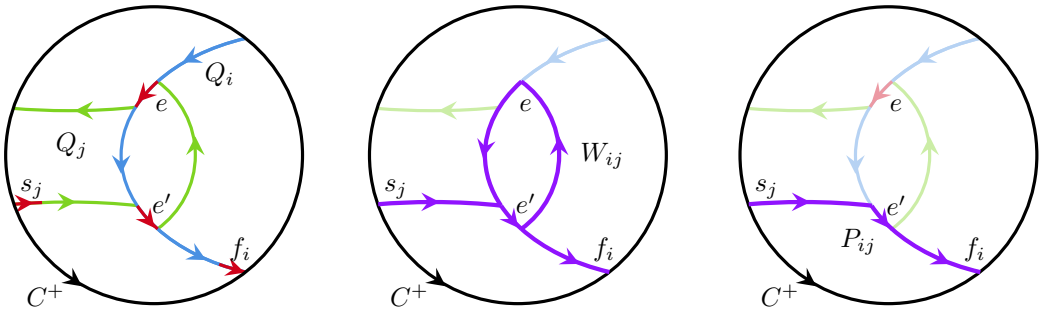


Figure 4 Possible configuration appearing in the proof of Lemma 16. Two jumps Q_i (blue) and Q_j (green) intersect in two different edges $e \in Q_1 \cap Q_2 \cap Q_3$ and $e' \in Q_i \cap Q_j$. The first edge of Q_j is denoted s_j and the last edge of Q_i is denoted f_i (red). The walk W_{ij} (purple, middle) from s_j to f_i is defined as $W_{ij} = Q_j[s_j, e] \cup Q_i(e, f_i]$ and uses the edge e' twice. The simple path P_{ij} (purple, right) is the simple sub-path from W_{ij} starting at s_j and ending at f_i.

path $P_{ij} \subseteq W_{ij}$ that goes from s_j to f_i. Let $C_{ij} = P_{ij} \cup C^+[r(Q_i), r(Q_j)]$. C_{ij} is a simple directed cycle since P_{ij} is a simple path from s_j to f_i outside of C^+ and $C^+[r(Q_i), r(Q_j)]$ is a simple path on C^+ from the endpoint of f_i on C^+ to the endpoint of s_j on C^+. An example of this construction is shown in Figure 5. If C_{ij} is a non-positive cycle, then we can replace C_{Q_i} and C_{Q_j} by C_{ij} in $\mathcal{C}$ and Conditions 1 and 2 would still hold. Indeed, by construction C_{ij} intersects C^+ on the single path $C^+[r(Q_i), r(Q_j)]$ so Condition 2 holds. Condition 1 holds because every edge $e \in E^+(C^+)$ that is contained in C_{Q_i} or C_{Q_j} is in fact in $r(Q_i)$ or $r(Q_j)$, which are both contained in C_{ij}. Hence it remains to show that there exists a pair $(i, j) \in \{(1, 2), (2, 3), (3, 1)\}$ such that C_{ij} is a non-positive cycle.

Suppose there exists a pair $(i, j) \in \{(1, 2), (2, 3), (3, 1)\}$ such that $r(Q_i)$ and $r(Q_j)$ are intersecting. Then we have:

$$\begin{aligned}
|E^+(C_{ij})| &= |E^+(P_{ij})| + |E^+(C^+[r(Q_i), r(Q_j)])| \\
&= |E^+(P_{ij})| + |E^+(r(Q_i) \cup r(Q_j))| \\
&\leq |E^+(W_{ij})| + |E^+(r(Q_i))| + |E^+(r(Q_j))| && \text{(because } P_{ij} \subseteq W_{ij}\text{)} \\
&= |E^+(Q_j[s_j, e])| + |E^+(Q_i(e, f_i])| + |E^+(r(Q_i))| + |E^+(r(Q_j))| \\
&\leq |E^+(Q_j)| + |E^+(Q_i)| + |E^+(r(Q_i))| + |E^+(r(Q_j))| \\
&= |E^+(C_{Q_i})| + |E^+(C_{Q_j})| \\
&< 2 \cdot \frac{1}{3}k && \text{(by Observation 10)}
\end{aligned}$$

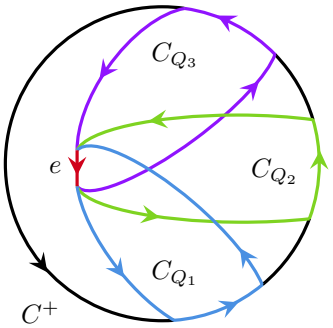
where the second equality follows from $C^+[r(Q_i), r(Q_j)] \subseteq r(Q_i) \cup r(Q_j)$. By Observation 10, C_{ij} cannot be positive and hence is a non-positive cycle, so we get the desired result.

If none of the paths $r(Q_1), r(Q_2), r(Q_3)$ are intersecting then assume for the sake of contradiction that C_{12}, C_{23} and C_{31} are all positive. By Observation 10 this means that

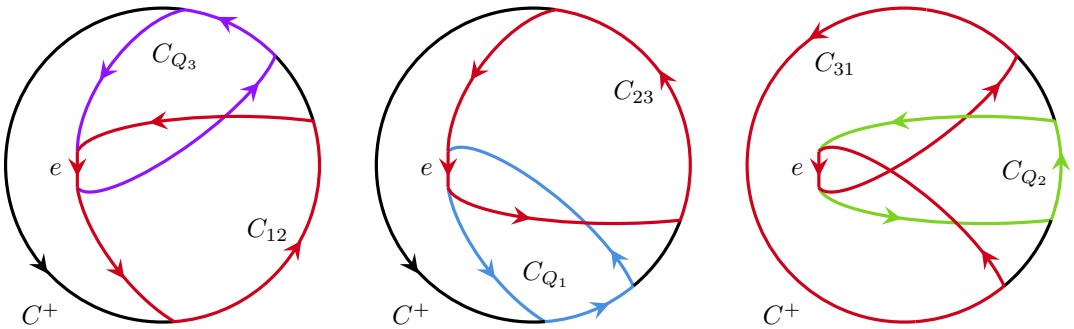
$$|E^+(C_{12})| + |E^+(C_{23})| + |E^+(C_{31})| > 3 \cdot \frac{2}{3}k = 2k.$$

However we also have for any pair $(i, j) \in \{(1, 2), (2, 3), (3, 1)\}$ that

$$\begin{aligned}
|E^+(C_{ij})| &= |E^+(P_{ij})| + |E^+(C^+[r(Q_i), r(Q_j)])| \\
&\leq |E^+(W_{ij})| + |E^+(C^+[r(Q_i), r(Q_j)])| && \text{(because } P_{ij} \subseteq W_{ij}\text{)} \\
&= |E^+(Q_j[s_j, e])| + |E^+(Q_i(e, f_i])| + |E^+(C^+[r(Q_i), r(Q_j)])|
\end{aligned}$$



(a) Three non-positive cycles C_{Q_1} (blue) C_{Q_2} (green) and C_{Q_3} (purple) intersecting in a single negative edge $e \notin C^+$.



(b) The newly constructed cycles C_{12}, C_{23} and C_{31} (in red from left to right). To construct C_{12} we combine the two cycles C_{Q_1} and C_{Q_2}.

Figure 5 Construction of three cycles C_{12}, C_{23} and C_{31} as done in the proof of Lemma 16. Note that this is a simplified configuration where the pairwise intersection of two non-positive cycles corresponds to the intersection of the three non-positive cycles C_{Q_1}, C_{Q_2} and C_{Q_3}.

Hence the total number of positive edges in the three cycles is:

$$\begin{aligned}
&|E^+(C_{12})| + |E^+(C_{23})| + |E^+(C_{31})| \\
&\quad \leq |E^+(Q_2[s_2, e])| + |E^+(Q_1(e, f_1])| + |E^+(C^+[r(Q_1), r(Q_2)])| \\
&\qquad + |E^+(Q_3[s_3, e])| + |E^+(Q_2(e, f_2])| + |E^+(C^+[r(Q_2), r(Q_3)])| \\
&\qquad + |E^+(Q_1[s_1, e])| + |E^+(Q_3(e, f_3])| + |E^+(C^+[r(Q_3), r(Q_1)])| \\
&\quad = |E^+(Q_1)| + |E^+(Q_2)| + |E^+(Q_3)| \\
&\qquad + |E^+(C^+)| + |E^+(r(Q_1))| + |E^+(r(Q_2))| + |E^+(r(Q_3))| \\
&\quad = |E^+(C^+)| + |E^+(C_{Q_1})| + |E^+(C_{Q_2})| + |E^+(C_{Q_3})| \\
&\quad < k + 3 \cdot \frac{1}{3}k = 2k \qquad \text{(by Observation 10)}
\end{aligned}$$

where we can apply Observation 10 because C_{Q_i} is non-positive for any $i \in \{1, 2, 3\}$. Hence there has to exists a pair $(i, j) \in \{(1, 2), (2, 3), (3, 1)\}$ such that C_{ij} is non-positive. ◀

3.5 Constructing a target set

We can now prove Lemma 9.

Proof of Lemma 9. For the sake of contradiction let (G, k, M^*, M) be a critical tuple. Apply Lemmas 13, 16 and 15 in that order consecutively and get a target set $\mathcal{C}$. By Lemma 11, this leads to a contradiction. Thus (G, k, M^*, M) cannot be a critical tuple. ◀

As a consequence, Lemmas 8 and 9 together show that Algorithm 1 is a polynomial-time algorithm that always outputs a perfect matching M containing between $\frac{1}{3}k$ and k red edges. This completes the proof of Theorem 1.

4 Conclusion

In this paper we formally define EM-OPT, an optimization variant of the EXACT MATCHING problem and show a deterministic polynomial-time approximation algorithm for EM-OPT which achieves an approximation ratio of 3 on bipartite graphs. Although the algorithm is fairly simple to present, the proof of its correctness is more complex. In the second part of the algorithm we iterate over all edges e and compute a perfect matching of minimum number of red edges containing e. Most of our work is put into proving that there exists such a matching that approximates the optimal solution of EM-OPT by a factor 3. As our calculations are tight for an approximation ratio of 3, a natural continuation of our work would be to improve this ratio, e.g. to 2. We speculate that the algorithm could be improved by iterating over larger subsets of edges (of constant size) instead of one edge at a time and that such an algorithm could give a better approximation and maybe even a PTAS. The analysis of such an algorithm, however, remains quite challenging and new techniques and insights might be needed. Another open problem is to find an approximation algorithm for general graphs.

References

1 Ilan Doron Arad, Ariel Kulik, and Hadas Shachnai. An EPTAS for budgeted matching and budgeted matroid intersection. *CoRR*, abs/2302.05681, 2023. doi:10.48550/arXiv.2302.05681.

2 Vikraman Arvind, Johannes Köbler, Sebastian Kuhnert, and Jacobo Torán. Solving linear equations parameterized by hamming weight. *Algorithmica*, 75(2):322–338, 2016. doi:10.1007/s00453-015-0098-3.

3 André Berger, Vincenzo Bonifaci, Fabrizio Grandoni, and Guido Schäfer. Budgeted matching and budgeted matroid intersection via the gasoline puzzle. *Math. Program.*, 128(1-2):355–372, 2011. doi:10.1007/s10107-009-0307-4.

4 Jacek Blazewicz, Piotr Formanowicz, Marta Kasprzak, Petra Schuurman, and Gerhard J. Woeginger. A polynomial time equivalence between DNA sequencing and the exact perfect matching problem. *Discret. Optim.*, 4(2):154–162, 2007. doi:10.1016/j.disopt.2006.07.004.

5 Paolo M. Camerini, Giulia Galbiati, and Francesco Maffioli. Random pseudo-polynomial algorithms for exact matroid problems. *J. Algorithms*, 13(2):258–273, 1992. doi:10.1016/0196-6774(92)90018-8.

6 Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of research of the National Bureau of Standards B*, 69(125-130):55–56, 1965.

7 Nicolas El Maalouly. Exact matching: Algorithms and related problems. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPIcs*, pages 29:1–29:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.STACS.2023.29.

8 Nicolas El Maalouly and Raphael Steiner. Exact matching in graphs of bounded independence number. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022, August 22-26, 2022, Vienna, Austria*, volume 241 of *LIPIcs*, pages 46:1–46:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.MFCS.2022.46.

9 Nicolas El Maalouly, Raphael Steiner, and Lasse Wulf. Exact matching: Correct parity and FPT parameterized by independence number. *CoRR*, abs/2207.09797, 2022. doi:10.48550/arXiv.2207.09797.

10 Dennis Fischer, Tim A. Hartmann, Stefan Lendl, and Gerhard J. Woeginger. An investigation of the recoverable robust assignment problem. In Petr A. Golovach and Meirav Zehavi, editors, *16th International Symposium on Parameterized and Exact Computation, IPEC 2021, September 8-10, 2021, Lisbon, Portugal*, volume 214 of *LIPIcs*, pages 19:1–19:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.IPEC.2021.19.

11 Anna Galluccio and Martin Loebl. On the theory of pfaffian orientations. I. perfect matchings and permanents. *Electron. J. Comb.*, 6, 1999. doi:10.37236/1438.

12 Hans-Florian Geerdes and Jácint Szabó. A unified proof for Karzanov's exact matching theorem. *quick proof QP-2011-02, Egerváry Research Group, Budapest*, 2011.

13 Fabrizio Grandoni and Rico Zenklusen. Optimization with more than one budget. *CoRR*, abs/1002.2147, 2010. arXiv:1002.2147.

14 Rohit Gurjar, Arpita Korwar, Jochen Messner, Simon Straub, and Thomas Thierauf. Planarizing gadgets for perfect matching do not exist. *ACM Trans. Comput. Theory*, 8(4):14:1–14:15, 2016. doi:10.1145/2934310.

15 Rohit Gurjar, Arpita Korwar, Jochen Messner, and Thomas Thierauf. Exact perfect matching in complete graphs. *ACM Trans. Comput. Theory*, 9(2):8:1–8:20, 2017. doi:10.1145/3041402.

16 AV Karzanov. Maximum matching of given weight in complete and complete bipartite graphs. *Cybernetics*, 23(1):8–13, 1987.

17 Monaldo Mastrolilli and Georgios Stamoulis. Constrained matching problems in bipartite graphs. In Ali Ridha Mahjoub, Vangelis Markakis, Ioannis Milis, and Vangelis Th. Paschos, editors, *Combinatorial Optimization – Second International Symposium, ISCO 2012, Athens, Greece, April 19-21, 2012, Revised Selected Papers*, volume 7422 of *Lecture Notes in Computer Science*, pages 344–355. Springer, 2012. doi:10.1007/978-3-642-32147-4_31.

18 Monaldo Mastrolilli and Georgios Stamoulis. Bi-criteria and approximation algorithms for restricted matchings. *Theor. Comput. Sci.*, 540:115–132, 2014. doi:10.1016/j.tcs.2013.11.027.

19 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Comb.*, 7(1):105–113, 1987. doi:10.1007/BF02579206.

20 Christos H. Papadimitriou and Mihalis Yannakakis. The complexity of restricted spanning tree problems. *J. ACM*, 29(2):285–309, April 1982. doi:10.1145/322307.322309.

21 Georgios Stamoulis. Approximation algorithms for bounded color matchings via convex decompositions. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 – 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, volume 8635 of *Lecture Notes in Computer Science*, pages 625–636. Springer, 2014. doi:10.1007/978-3-662-44465-8_53.

22 Ola Svensson and Jakub Tarnawski. The matching problem in general graphs is in quasi-NC. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 696–707. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.70.

23 Moshe Y Vardi and Zhiwei Zhang. Quantum-inspired perfect matching under vertex-color constraints. *arXiv preprint arXiv:2209.13063*, 2022.

24 Tongnyoul Yi, Katta G. Murty, and Cosimo Spera. Matchings in colored bipartite networks. *Discret. Appl. Math.*, 121(1-3):261–277, 2002. doi:10.1016/S0166-218X(01)00300-6.

25 Raphael Yuster. Almost exact matchings. *Algorithmica*, 63(1-2):39–50, 2012. doi:10.1007/s00453-011-9519-0.

A Proof of Lemma 12

This section is devoted to the proof of Lemma 12. The proof technique is a combination of reasonings about forward and backward motion, shown in Lemmas 17 and 18, and an interpolation argument.

Let's start with a basic fact about sub-paths of C_e. Let e_s, e_f be two distinct edges on C^+. The following lemma captures the intuitive fact that any path from e_s to e_f must go either in a clockwise motion, or in an anticlockwise motion. While doing so, it must traverse all the edges between e_s and e_f in that direction.

▶ **Lemma 17.** *Let (G, k, M^*, M) be a critical tuple and fix an edge $e \in E^+(C^+)$. If e_s, e_f are distinct edges in C^+ and $\mathcal{P} = C_e[e_s, e_f]$ is a sub-path of C_e from e_s to e_f, then either $C^+[e_s, e_f] \subseteq r_f(\mathcal{P})$ or $C^+[e_f, e_s] \subseteq r_b(\mathcal{P})$.*

Proof. As a helpful tool, we consider the following definition of *shadow*. Let Q be a jump of C_e from x to y, where x and y are vertices in C^+. If Q is a forward jump, then the shadow $s(Q)$ of Q is the walk in C^+, which starts at x and goes anticlockwise, until it encounters y. If Q is a backward jump, then $s(Q)$ is the walk in C^+, which starts at x and goes clockwise, until it encounters y. We remark that formally, since $s(Q)$ can travel edges in reverse direction, $s(Q)$ is a walk in the undirected analogue of C^+ in G. For an interjump P from vertex x to y, the shadow of P is defined to be the path P itself. For a sub-path $\mathcal{P} \subseteq C_e$ consisting of the concatenation of alternating jumps and interjumps the shadow $s(\mathcal{P})$ of $\mathcal{P}$ is the concatenation of the corresponding shadows of jumps and interjumps in the same order.

By definition, the following basic observations hold. For any sub-path $\mathcal{P} \subseteq C_e$, its shadow $s(\mathcal{P})$ is a walk contained in C^+ such that it has the same start and end vertex as $\mathcal{P}$. Furthermore, the walk $s(\mathcal{P})$ can be thought of as a sequence of steps, where every step traverses one single edge of C^+ either clockwise or anticlockwise. The set $r_f(\mathcal{P})$ is exactly the set of edges which are traveled by the walk $s(\mathcal{P})$ with an anticlockwise step. The set $r_b(\mathcal{P})$ is exactly the set of edges which are traveled by the walk $s(\mathcal{P})$ with a clockwise step. The set $r(\mathcal{P})$ is exactly the set of edges which are traveled by the walk $s(\mathcal{P})$.

Now return to the statement of the lemma. The walk $s(\mathcal{P})$ is a walk contained in C^+, starting with the edge e_s and ending with the edge e_f. It is obvious that $s(\mathcal{P})$ must either traverse all of $C^+[e_s, e_f]$ with anticlockwise steps, or all of $C^+[e_f, e_s]$ with clockwise steps. This implies that either $C^+[e_s, e_f] \subseteq r_f(\mathcal{P})$ or $C^+[e_f, e_s] \subseteq r_b(\mathcal{P})$. ◀

In the proof of Lemma 12, we will consider a set of forward jumps and interjumps in C_e that cover all the cycle C^+. However we want to avoid forward jumps covering the edge e. The following Lemma 18 shows that this is possible under certain assumptions. Intuitively, since e is already contained in an interjump of C_e, there is no need to include a forward jump covering e in the set of forward jumps and interjumps covering C^+. However one needs to carefully select the forward jumps of the set, which explains the technicallity of the proof below.

▶ **Lemma 18.** *Let (G, k, M^*, M) be a critical tuple and fix an edge $e \in E^+(C^+)$. If $e \notin r_b(C_e)$, then there exists a set of forward jumps $\mathcal{Q} \subseteq \mathcal{J}_e^f$ such that*

$$C^+ = \bigcup_{Q \in \mathcal{Q}} r(Q) \cup \bigcup_{P \in \mathcal{I}_e} P$$

and $\nexists Q_f \in \mathcal{Q}$ with $e \in r(Q_f)$.

Proof. We prove the statement by constructing a tuple $(\mathcal{P}, \mathcal{R}, e')$, where $e' \in C^+$ is an edge of C^+, $\mathcal{P} = C_e[e, e']$ is a sub-path of C_e and $\mathcal{R}$ is a set of jumps and interjumps of C_e. We want $(\mathcal{P}, \mathcal{R}, e')$ to satisfy the following invariant.

▶ **Invariant.** *A tuple $(\mathcal{P}, \mathcal{R}, e')$ satisfies the invariant if the following properties hold:*

1. *There is no forward jump Q in $\mathcal{R}$ such that $e \in r(Q)$.*
2. $C^+[e, e'] \subseteq r_f(\mathcal{P})$.
3. $C^+(e', e) \subseteq r_f(\mathcal{R})$

We also want to ensure that there is no forward jump Q in $\mathcal{P}$ with $e \in r(Q)$. Indeed, in that case, define $\mathcal{Q}$ to be the set of forward jumps in $\mathcal{P} \cup \mathcal{R}$. Then by Invariant 1, $\mathcal{Q}$ doesn't contain any forward jump Q with $e \in r(Q)$. Also, by Invariants 2 and 3 of $(\mathcal{P}, \mathcal{R}, e')$:

$$\begin{aligned} C^+ &= C^+[e, e'] \cup C^+(e', e) \\ &\subseteq r_f(\mathcal{P}) \cup r_f(\mathcal{R}) \\ &\subseteq \bigcup_{Q \in \mathcal{Q}} r(Q) \cup \bigcup_{P \in \mathcal{I}_e} P \end{aligned}$$

which is the conclusion of Lemma 18.

We start with a tuple $(\mathcal{P}, \mathcal{R}, e')$ satisfying the invariant properties and then iteratively modify the tuple in order to achieve the additional property that no forward jump in $\mathcal{P}$ covers e. At first, e' is the edge before e in C_e. Note that $e' \in C^+$ because $e \notin M$ and C_e is M-alternating so $e' \in M$, and since C^+ is also M-alternating, edges of M adjacent to e must be in C^+. Let $\mathcal{P} = C_e[e, e']$ be a path starting in e and covering all edges of C_e and let $\mathcal{R} = \emptyset$ be an empty set. Clearly Invariants 1 and 3 holds for $(\mathcal{P}, \mathcal{R}, e')$. By Lemma 17 applied on $\mathcal{P}$, either $C^+[e, e'] \subseteq r_f(\mathcal{P})$ or $C^+[e', e] \subseteq r_b(\mathcal{P})$. However $e \notin r_b(C_e)$ so the latter is not possible. Hence $(\mathcal{P}, \mathcal{R}, e')$ satisfies all invariant properties. Let t be the number of forward jumps Q in $\mathcal{P}$ with $e \in r(Q)$. If $t = 0$, we are done.

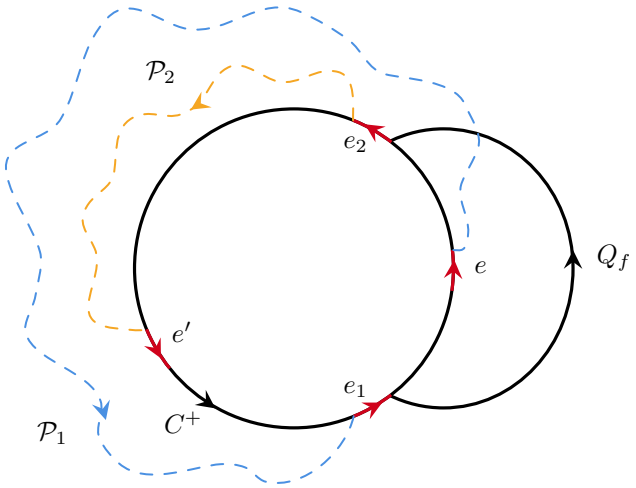


Figure 6 To prove Lemma 18 we consider a path $\mathcal{P} = C_e[e, e']$ that contains a jump Q_f with $e \in r(Q_f)$. e_1 and e_2 are the edges before and after the jump Q_f. We split $\mathcal{P}$ into two paths $\mathcal{P}_1$ (blue) and $\mathcal{P}_2$ (orange) that are respectively before and after the jump Q_f.

If $t > 0$, we replace $(\mathcal{P}, \mathcal{R}, e')$ by another tuple $(\mathcal{P}', \mathcal{R}', e_1)$ where $\mathcal{P}'$ contains $t-1$ forward jumps covering e. We ensure that $(\mathcal{P}', \mathcal{R}', e_1)$ also satisfies the invariant properties, so that we can repeat the process until $t = 0$. Let Q_f be the last forward jump in $\mathcal{P}$ covering e

and let e_1 and e_2 be the first edges in C^+ before and after Q_f (see Figure 6). Consider the sub-paths $\mathcal{P}_1 = C_e[e, e_1]$ and $\mathcal{P}_2 = C_e[e_2, e']$ and define $\mathcal{P}' := \mathcal{P}_1$ and $\mathcal{R}' := \mathcal{R} \cup \mathcal{P}_2$. By definition, $\mathcal{P}'$ is the sub-path of $\mathcal{P}$ before Q_f (excluded), hence it contains $t-1$ forward jumps covering e. It remains to show that $(\mathcal{P}', \mathcal{R}', e_1)$ satisfies the invariant. By definition of Q_f, there is no forward jump in $\mathcal{P}_2$ covering e. Thus, since $(\mathcal{P}, \mathcal{R}, e')$ satisfies Invariant 1, $(\mathcal{P}', \mathcal{R}', e_1)$ also satisfies Invariant 1. We can apply Lemma 17 on $\mathcal{P}_1$ and get that either $C^+[e, e_1] \subseteq r_f(\mathcal{P}_1)$ or $C^+[e_1, e] \subseteq r_b(\mathcal{P}_1)$. However $e \notin r_b(C_e)$ so the latter case is not possible, hence $(\mathcal{P}', \mathcal{R}', e_1)$ satisfies Invariant 2. Finally, to prove that $C^+(e_1, e) \subseteq r_f(\mathcal{R}')$, consider two cases. If $C^+(e_1, e) \subseteq C^+(e', e)$ (as in Figure 6) then $C^+(e_1, e) \subseteq r_f(\mathcal{R}) \subseteq r_f(\mathcal{R}')$ by Invariant 3 of $(\mathcal{P}, \mathcal{R}, e')$. Otherwise if $C^+(e', e) \subseteq C^+(e_1, e)$, then the edges e_2, e_1, e', e appear in that order on C^+. We can thus write $C^+(e_1, e) = C^+(e_1, e'] \cup C^+(e', e)$. By Invariant 3 of $(\mathcal{P}, \mathcal{R}, e')$, we know that $C^+(e', e) \subseteq r_f(\mathcal{R})$. Note that $C^+(e_1, e'] \subseteq C^+[e_2, e']$. Applying Lemma 17 on $\mathcal{P}_2$ we get that either $C^+[e_2, e'] \subseteq r_f(\mathcal{P}_2)$ or $C^+[e', e_2] \subseteq r_b(\mathcal{P}_2)$. But $e \in C^+[e', e_2]$ and $e \notin r_b(C_e)$ so the latter case is not possible. Hence $C^+(e_1, e'] \subseteq \mathcal{P}_2$ and we thus have $C^+(e_1, e) \subseteq r_f(\mathcal{P}_2) \cup r_f(\mathcal{R}) = r_f(\mathcal{R}')$. So in both cases Invariant 3 holds for $(\mathcal{P}', \mathcal{R}', e_1)$. Therefore, we can safely replace $(\mathcal{P}, \mathcal{R}, e')$ by $(\mathcal{P}', \mathcal{R}', e_1)$ and maintain the invariant. ◀

We now have collected all the ingredients to prove the main result of this subsection. While the previous lemmas were proven using an intuition on forward and backward motion, Lemma 12 is proven using an interpolation argument.

Proof of Lemma 12. For the sake of contradiction, assume that there exists an edge $e \in E^+(C^+)$ such that $e \notin r_b(C_e)$. Apply Lemma 18 and let $\mathcal{Q}'$ be the resulting set of forward jumps. By identifying every jump in $\mathcal{Q}'$ with its reach, we can apply Observation 14 and get the resulting set $\mathcal{Q} \subseteq \mathcal{Q}'$. Consequently, $C^+ = \bigcup_{Q \in \mathcal{Q}} r(Q) \cup \bigcup_{P \in \mathcal{I}_e} P$, there is no jump $Q \in \mathcal{Q}$ covering e and every edge $e' \in C^+$ is contained in the reach of at most two jumps of $\mathcal{Q}$. Order the jumps in $\mathcal{Q}$ by the distance between e and their reach and write $\mathcal{Q} = \{Q_1, Q_2, \dots, Q_t\}$. Define $\mathcal{Q}_{odd} = \{Q_1, Q_3, \dots\}$ and $\mathcal{Q}_{even} = \{Q_2, Q_4, \dots\}$ to be the set of jumps of respectively odd and even index. Since there is no forward jump in $\mathcal{Q}$ covering e, the first and the last jump in $\mathcal{Q}$ are non-intersecting (otherwise the reach of one of them would contain e). Additionally, the reach of a jump Q_i can only intersect the reach of Q_{i-1} and Q_{i+1} (by the property obtained by Observation 14). Hence the reach of any two jumps in $\mathcal{Q}_{odd}$ as well as the reach of any two jumps in $\mathcal{Q}_{even}$ are not intersecting.

We can thus define two cycles $C^+_{odd} = C^+ \cup \mathcal{Q}_{odd} \setminus \{r(Q_1), r(Q_3), \dots\}$ and $C^+_{even} = C^+ \cup \mathcal{Q}_{even} \setminus \{r(Q_2), r(Q_4), \dots\}$. We show that at least one of them is non-positive using Observation 10. Indeed, the number of positive edges in C^+_{odd} is

$$\begin{aligned} |E^+(C^+_{odd})| &= |E^+(C^+)| + \sum_{Q \in \mathcal{Q}_{odd}} |E^+(Q)| - \sum_{Q \in \mathcal{Q}_{odd}} |E^+(r(Q))| \\ &= |E^+(C^+)| - \sum_{Q \in \mathcal{Q}_{odd}} m_Q \end{aligned}$$

where we define $m_Q := |E^+(r(Q))| - |E^+(Q)|$. Assume for now that $\sum_{Q \in \mathcal{Q}_{odd}} m_Q \geq \sum_{Q \in \mathcal{Q}_{even}} m_Q$ so that $\sum_{Q \in \mathcal{Q}} m_Q = \sum_{Q \in \mathcal{Q}_{odd}} m_Q + \sum_{Q \in \mathcal{Q}_{even}} m_Q \leq 2 \sum_{Q \in \mathcal{Q}_{odd}} m_Q$. So we get

$$|E^+(C^+_{odd})| \leq |E^+(C^+)| - \frac{1}{2} \sum_{Q \in \mathcal{Q}} m_Q$$

and we also have

$$\begin{aligned}
\sum_{Q\in\mathcal{Q}} m_Q &= \sum_{Q\in\mathcal{Q}} |E^+(r(Q))| - \sum_{Q\in\mathcal{Q}} |E^+(Q)| \\
&= \left(\sum_{Q\in\mathcal{Q}} |E^+(r(Q))| + \sum_{P\in\mathcal{I}_e} |E^+(P)|\right) - \left(\sum_{P\in\mathcal{I}_e} |E^+(P)| + \sum_{Q\in\mathcal{Q}} |E^+(Q)|\right) \\
&\geq |E^+(C^+)| - \left(\sum_{P\in\mathcal{I}_e} |E^+(P)| + \sum_{Q\in\mathcal{Q}} |E^+(Q)|\right) \qquad \text{(by the construction of } \mathcal{Q}) \\
&\geq |E^+(C^+)| - |E^+(C_e)|
\end{aligned}$$

where the last inequality comes from the fact that C_e can contain jumps outside of $\mathcal{Q}$. We finally get the following bound on the number of positive edges in C^+_{odd}.

$$\begin{aligned}
|E^+(C^+_{odd})| &\leq |E^+(C^+)| - \frac{1}{2}(|E^+(C^+)| - |E^+(C_e)|) \\
&= \frac{1}{2}|E^+(C^+)| + \frac{1}{2}|E^+(C_e)| \\
&< \frac{1}{2}k + \frac{1}{2}\cdot\frac{1}{3}k \qquad \text{(by Observation 10)} \\
&= \frac{2}{3}k
\end{aligned}$$

By Observation 10, we can conclude that C^+_{odd} is a non-positive cycle. If $\sum_{Q\in\mathcal{Q}_{odd}} m_Q \leq \sum_{Q\in\mathcal{Q}_{even}} m_Q$ then the same argument on C^+_{even} shows that C^+_{even} is a non-positive cycle. In the following we assume that C^+_{odd} is a non-positive cycle, but the reasoning can be adapted for the case that C^+_{even} is non-positive by interchanging *odd* by *even*.

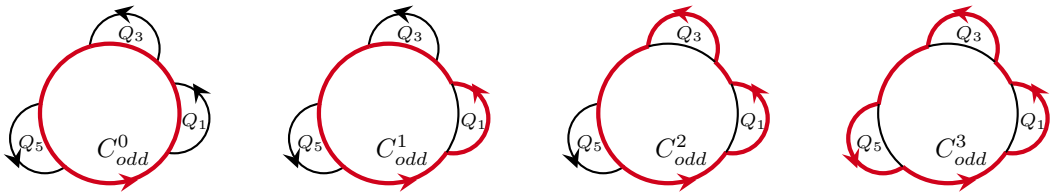


Figure 7 A sequence of cycles $C^0_{odd}, C^1_{odd}, C^2_{odd}, C^3_{odd}$ (in red from left to right) defined by sequentially adding the jumps in $\mathcal{Q}_{odd} = \{Q_1, Q_3, Q_5\}$ to the cycle C^+ and removing the corresponding reach, as done in the proof of Lemma 12. Note that $C^0_{odd} = C^+$.

Define a sequence of cycles $C^{i+1}_{odd} := C^i_{odd} \cup Q_{2i+1} \setminus r(Q_{2i+1})$ for $i \in |\mathcal{Q}_{odd}|$ starting at $C^0_{odd} := C^+$ and ending at C^+_{odd}. See Figure 7 for an example. The constructed C^i_{odd}'s are simple cycles since the reaches in $\mathcal{Q}_{odd}$ are non-intersecting. The sequence starts with a positive cycle C^+ and ends with a non-positive cycle C^+_{odd} so there must exists an index i such that C^i_{odd} is positive and C^{i+1}_{odd} is non-positive. However the number of positive edges in the non-positive cycle C^{i+1}_{odd} is

$$\begin{aligned}
|E^+(C^{i+1}_{odd})| &= |E^+(C^i_{odd})| + |E^+(Q_{2i+1})| - |E^+(r(Q_{2i+1})|) \\
&= |E^+(C^i_{odd})| + |E^+(C_{Q_{2i+1}})| - |E^+(C^+)| \\
&> \frac{2}{3}k + \frac{2}{3}k - k \qquad \text{(by Observation 10)} \\
&= \frac{1}{3}k
\end{aligned}$$

By Observation 10, this implies that C^{i+1}_{odd} is positive, but we previously argued that it is non-positive. Hence it is not possible that $e \notin r_b(C_e)$. ◀