# Recovery of disrupted airline operations using k-Maximum Matching in Graphs*

Nicolas Nisse[1], Alexandre Salch[2], and Valentin Weber[3]

[1] Université Côte d'Azur, Inria, CNRS, I3S, France
[2] Innovation & Research, Amadeus IT Group SA
[3] Innovation & Research, Amadeus IT Pacific

**Abstract.** When an aircraft is approaching an airport, it gets a short time interval called a *slot* by Air Traffic Control that it can use to land. If the landing of the aircraft is delayed for some reason, it loses its slot and Airline Operation Controllers have to assign it a new one. However, landing slots are a scarce resource of the airports and, in order to avoid that an aircraft waits too long, Airline Operation Controllers have to regularly modify the assignment of slots to the aircraft. Due to the system implemented to exchange slots, controllers can modify the slot-assignment using only two kinds of operations: either assign a slot that was free to aircraft $A$, or give the slot of another aircraft $B$ to $A$ and assign a free slot to $B$. The problem can then be modeled as follows.
Let $k \geq 1$ be an odd integer, let $G$ be a graph and $M$ be a matching in $G$, i.e. a set of pairwise disjoint edges. A $k$-maximum matching is a largest possible matching that can be obtained from $M$ by using only augmenting paths of length at most $k$. This problem has already been studied in the context of wireless networks, mainly because it provides a simple approximation for the maximum matching problem. This paper provides a polynomial-time algorithm when $k \leq 3$. We then prove that the problem is NP-hard in planar bipartite graphs with maximum degree at most 3 for any odd integer $k \geq 5$.

**Keywords:** Graph Theory; Matching; Augmenting paths; Complexity

## 1 Introduction

In regions where air traffic is dense, authorities are putting in place regulations to better share the available resources, such as airport runways or routes. In the case of runways, this exclusive resource can be shared across aircraft by issuing airport landing slots. Aircraft that operate those flights have to land within the time interval as defined by the slot, usually 20 minutes. The slots considered in this article are obtained at the operational phase that is up to a couple of days before departure and are not the slots obtained during the planning phase [Sie10]. These slots model the real time capacity of an airport and their rate may decrease should disruptions such as bad weather conditions occur.

Air Traffic Control (ATC) only authorises the aircraft to take off at the departure airport if the flight plan ensures that the aircraft will land within its slot. The purpose of this procedure is to clear the air space around the destination airport by having less aircraft waiting for landing and also to save fuel costs. Indeed, aircraft will wait for clearance to take off at the departure airport instead of waiting for the authorization to land while circling over the destination airport. A report from the Australian ATC shows that the regional implementation is saving more than 180 M AUD a year from fuel savings while generating an additional 13 M AUD in crew costs and 43 M AUD in passenger costs due to higher total delays [Aus14].

In order to efficiently assign slots to flights, such a system implements the principles of Collaborative Decision Making (CDM). In CDM the most up to date information about the flights and the capacity of the controlled airports are gathered and made available to all stakeholders [Wam96]. Implementations of this principle differ from one region to another. In Europe, slots are allocated two hours before the expected departure time. The system's objective function is to assure equity of delay between different

* This work was partly funded by the ANR project STINT, and promoted by the Inria associate-team AlDyNet. This work is the extended version of half of the results presented in LAGOS 2017 [BGN+17].

airline operators. After this first allocation, airlines can still swap slots between their flights in order to minimize delay associated to high revenue flights [Eur12]. Another approach is implemented in Australia where slots are assigned to flights the day before the operations on a first come, first served basis with respect to the expected arrival times of the flights, i.e., the first flight is assigned to the best available slot, then the next flight to the best remaining slot and so on [Aus15a]. Another difference is that in the Australian implementation, airlines can exchange one of their assigned slots with an unassigned slot but also with a slot belonging to another airline by using a web interface [Aus15b]. These changes are subject to business rules ensuring that the other airlines are notified far enough in advance and that their flights are reassigned to better slots than their original slots. But no acknowledgement is needed from the other airlines. The main difference between the two implementations is that the European implementation is a centralized optimization system whereas the Australian implementation consists of multiple agents competing for the best slot allocation. As a consequence, knowing a procedure to optimally exchange slots can be a competitive advantage for an airline operating in this region. This paper focuses on the latter implementation.

## 2 Problem description

**Recovery of disrupted airline operations.** Airports have time intervals called *slots* that are initially assigned to aircraft according to their arrival schedule. An aircraft can only be assigned to a slot that is compatible with its arrival time. However, if a flight is delayed, airline operation controllers must assign a new slot via the information system managing these exchanges. The strict regulations of this system imply that only two operations (or rules) are possible. Either an available slot is assigned to the aircraft $A$ (**Rule** 1), or a slot $S$ which is already assigned to another aircraft $B$ is reassigned to $A$ while $B$ is assigned an available slot $S'$ (**Rule** 2). Note that Rule 2 corresponds to swapping the slots of $A$ and $B$ if $S'$ was previously assigned to $A$. In both cases, the slot $S$ should be compatible with the schedule of $A$ and in the second case, $S$ and $S'$ must be compatible with the schedule of $B$. If several planes fall behind and lose their slots, the resolution of these problems is difficult for airline operation controllers that do not have the tools to make these changes and must ensure manually that all aircraft will have a slot.

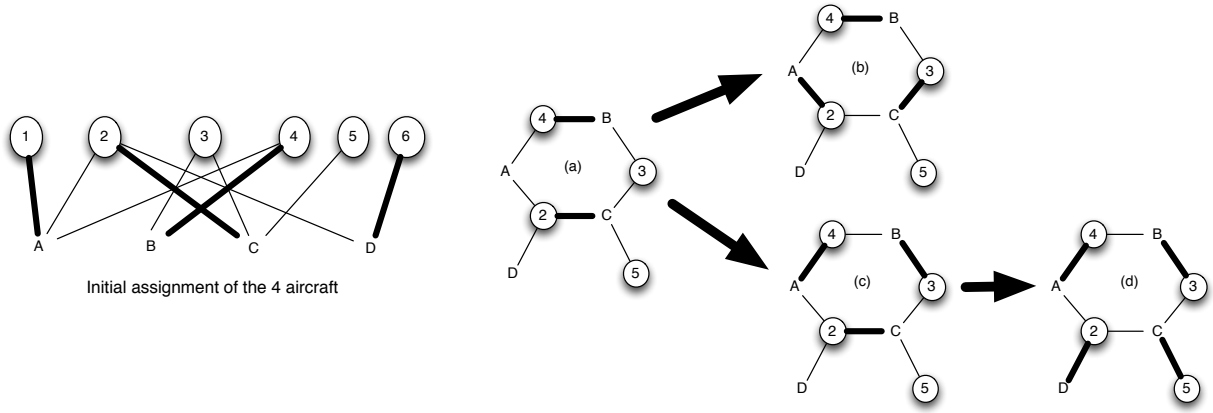



**Fig. 1.** Example of a simple scenario with 4 aircraft and 6 slots.

A simplified scenario is shown in Figure 1 with 4 planes denoted by $A, B, C, D$ and 6 slots denoted by $1, \ldots, 6$. On the left, each aircraft is linked to the slots that are compatible with it. Initially, slots $1, 2, 4$, and 6 are assigned to aircraft $A, C, B$, and $D$, respectively. Edges of the matching are depicted in bold. Assume that, after aircraft $A$ and $D$ have been delayed, they are not compatible with slots 1 and 6 anymore. Hence,

the configuration becomes the one depicted in Figure 1(a). From Configuration (a), if Slot 2 is reassigned to Plane $A$ and Slot 3 is assigned to Plane $C$ (i.e., Rule 2 is applied to Planes $A$ and $C$), then we reach Configuration ($b$) where no allowed modification is possible anymore (none of the Rules 1 or 2 can be applied). Another solution would be to apply Rule 2 to Planes $A$ and $B$ (Plane $A$ gets Slot 4 and Plane $B$ gets Slot 3), reaching Configuration ($c$), and then to apply Rule 2 to Planes $C$ and $D$ (Plane $C$ gets Slot 5 and Plane $D$ gets Slot 2), reaching the Configuration of Figure 1(d) where all aircraft are assigned a slot.

**Matching in Graphs.** The problem of *reassignment of slots* can be modelled as a problem of matching in graphs. Let $G = (V, E)$ be a graph. A *matching* $M \subseteq E$ of $G$ is a set of pairwise disjoint edges. A vertex $v \in V$ is *covered* by $M$ if there is $e \in M$ such that $v \in e$. Otherwise, $v$ is said to be *exposed*. The maximum size of a matching in $G$ is denoted by $\mu(G)$. The problem of computing a maximum matching has been widely studied in the literature and can be solved in polynomial time [Edm65]. A key ingredient in most of the work on matching is the notion of *augmenting paths*. A path $P = (v_0, \cdots, v_k)$ in $G$ is a sequence of pairwise distinct vertices such that $e_i = \{v_i, v_{i+1}\} \in E$ for each $0 \leq i < k$. The *length* of a path is its number of edges and a path is said to have *odd length* if its length is odd. The path $P$ is said to be $M$-augmenting if $v_0$ and $v_k$ are exposed and, for any $0 \leq i < k$, $e_i = \{v_i, v_{i+1}\} \in M$ if and only if $i$ is odd. In particular, an augmenting path always has an odd length (i.e., $k$ is odd since $v_k$ must be exposed). A theorem of Berge [Ber57] states that a matching $M$ is maximum if and only if there are no $M$-augmenting paths. In particular, if $P$ is $M$-augmenting, then $M \Delta E(P)$ is a matching of $G$ of size $|M| + 1$ where $E(P)$ is the set of edges of $P$ and $\Delta$ is the symmetric difference. When passing from a matching $M$ to the matching $M \Delta E(P)$, we say that the $M$-augmenting path $P$ is *augmented*.

The problem of *reassigning slots* described above can be modelled as follows. Let $G = (X \cup Y, E)$ be a bipartite graph. $X$ represents the set of aircraft and $Y$ represents the set of available slots. There is an edge between $a \in X$ and $s \in Y$ if and only if the slot $s$ is compatible with the schedule of the aircraft $a$. Let $M$ be a matching of $G$ that corresponds to a pre-established valid assignment of some slots to some aircraft (see Fig 1 (a)). The problem of reassignment of slots is equivalent to computing a maximum matching that can be obtained from $M$ by augmenting only paths of length at most 3. For instance, the first scenario in the above example (passing from Configuration ($a$) to ($b$)) consists in augmenting the path ($A2C3$). Reaching Configuration ($b$), there are no augmenting paths of length at most 3. The second scenario consists in augmenting first the path ($A4B3$) (reaching Configuration ($c$)) and then the path ($D2C5$).

**Related work.** The problem of finding a maximum matching in bipartite graphs has been extensively studied, in particular because it is a special case of a network flow problem. For example, Kuhn [Kuh55] showed that it can be solved in polynomial time using the Hungarian method. The first algorithm for solving the maximum matching problem in polynomial time in general graphs is due to Edmonds [Edm65]. Since then, much research has been done to design more efficient algorithms as in [HK73, MV80, DP14]. In particular, the algorithms proposed by Hopcroft and Karp [HK73] and by Micali and Vazirani [MV80] are based on augmenting paths in the non-decreasing order of their lengths. Such a method gives a good approximation since augmenting only the paths of length at most $2k-3$ provides a $(1-1/k)$-approximation of the maximum matching [HK73].

The problem of matching with bounded-length paths has also been studied in the context of wireless networks. In particular, it provides simple distributed algorithms to compute the scheduling of transmissions with interference [WS05, BSS09].

**Results summary.** Let $k$ be an odd integer. We consider the problem that takes a graph $G = (V, E)$ and a matching $M \subseteq E$ as inputs. Let $\mu_k(G, M)$ denote the size of a maximum matching that can be obtained from $M$ in $G$ by augmenting paths of length at most $k$. The *$k$-Matching Problem* consists of computing a sequence of augmenting paths of length $\leq k$ that allow to obtain, starting from $M$, a matching of size $\mu_k(G, M)$ (see formal definition in Section 3). Let us emphasize that only odd-length paths can be augmented.

The remainder of this paper is organized as follows. Section 3 introduces notation and preliminary results that will be referred to throughout the paper. Section 4 presents polynomial-time algorithms to solve special cases of this problem. In particular, Section 4.1 describes polynomial-time algorithms that compute $\mu_k(G, M)$

for any graph $G$ and starting from any matching $M$ for $k \in \{1, 3\}$. Then, in Section 4.2, we design a linear-time algorithm that computes $\mu_k(P, M)$ for any odd $k \geq 1$, and any path $P$ with initial matching $M$. Section 5 proves that the problem of computing $\mu_k(G, M)$ is NP-hard in planar bipartite graphs with maximum degree at most 3 for any odd integer $k \geq 5$. Finally, Section 6 presents simulations of the execution of the algorithm described in Section 4.1 on instances coming from the industry.

## 3 Preliminaries

This section is devoted to the formal definition of the $k$-Matching Problem and to some preliminary remarks. In particular, the case $k = 1$ is almost trivially equivalent to the Maximum Matching Problem from a computational complexity point of view.

Let $k \geq 1$ be an odd integer. The $k$-Matching Problem takes a graph $G = (V, E)$ and a matching $M \subseteq E$ as inputs. It aims at computing a matching $M^* \subseteq E$ that can be obtained from $M$ by sequentially augmenting paths of length at most $k$, and $M^*$ has maximum size subject to this constraint.

More formally, starting from the initial matching $M = M_0$, the problem consists of computing a sequence $(P_0, \cdots, P_r)$ of paths, each of length at most $k$, such that for any $0 \leq i \leq r$, $P_i$ is an $M_i$-augmenting path and $M_{i+1} = M_i \Delta E(P_i)$ (the symmetric difference between $M_i$ and the edges of $P_i$). Moreover, the size of the final matching $M_{r+1}$ must be maximum under these constraints. Note that $|M_i| = |M| + i$ for every $0 \leq i \leq r + 1$. Hence, the problem can be equivalently defined as the computation of a longest sequence of augmenting paths of length at most $k$ (starting from matching $M$ in graph $G$).

Let $\mu_k(G, M)$ denote the maximum size of a solution for the $k$-Matching Problem in the graph $G$, starting from the matching $M$. That is, $\mu_k(G, M)$ is the maximum size of a matching that can be obtained from $M$ by sequentially augmenting paths of length at most $k$. By previous paragraph, $\mu_k(G, M) - |M|$ is the maximum length of such a sequence of augmenting paths.

In what follows, for any $M \subseteq E$, let $V(M)$ be the set of vertices incident to some edge in $M$. For any $S \subseteq V$, let $G[S]$ denote the subgraph induced by $S$. Moreover, recall that $\mu(G)$ denotes the size of a (classical) maximum matching in $G$.

*Claim.* Let $G$ be any $n$-node graph and $M \subseteq E(G)$ be a matching, $\mu_{n-1}(G, M) = \mu(G)$.

*Proof.* Since any augmenting path has length at most $n-1$, this follows from Berge's Theorem [Ber57] which states that a matching is maximum if and only if there exist no augmenting paths. ◇

*Claim.* Let $k \geq 1$ be any odd integer and $G$ be any graph, $\mu_k(G, \emptyset) = \mu(G)$.

*Proof.* It is sufficient to start from a (classical) maximum matching $M$ of $G$ and then to augment all edges of $M$ one by one. That is, let $M = \{e_1, \cdots, e_r\}$ be a maximum matching in $G$ (that can be computed in polynomial time). For any $1 \leq i \leq r$, let $P_i$ be the path with $e_i$ as its unique edge. Augmenting the sequence of paths $(P_1, \cdots, P_r)$ leads to an optimal solution. ◇

Before going on, let us emphasize a property that will be extensively used throughout the paper.

*Property 1.* Let $G = (V, E)$ be a graph, $M \subseteq E$ be a matching, and $P$ be any $M$-augmenting path. If a vertex $v \in V$ is covered by $M$, then it is covered by the matching $M \Delta E(P)$ obtained after augmenting $P$.

Now, let us consider the case $k = 1$.

*Claim.* Let $G = (V, E)$ be any graph and $M \subseteq E$ be a matching. $\mu_1(G, M) = |M| + \mu(G[V \setminus V(M)])$

*Proof.* For $k = 1$, only edges between two exposed vertices can be augmented, i.e. between two vertices that are initially exposed by $M$ using Property 1. In particular, all edges of $M$ will belong to any matching $M_i$ obtained from $M$ by augmenting paths of length at most 1.

Let $H = G[V \setminus V(M)]$ be the graph obtained from $G$ by removing all vertices of the edges of $M$. Let $M'$ be a (classical) maximum matching of $H$. Then, $M \cup M'$ is a matching of $G$ with size $\mu_1(G, M)$ that can be obtained from $M$ by augmenting all edges of $M'$ one by one. ◇

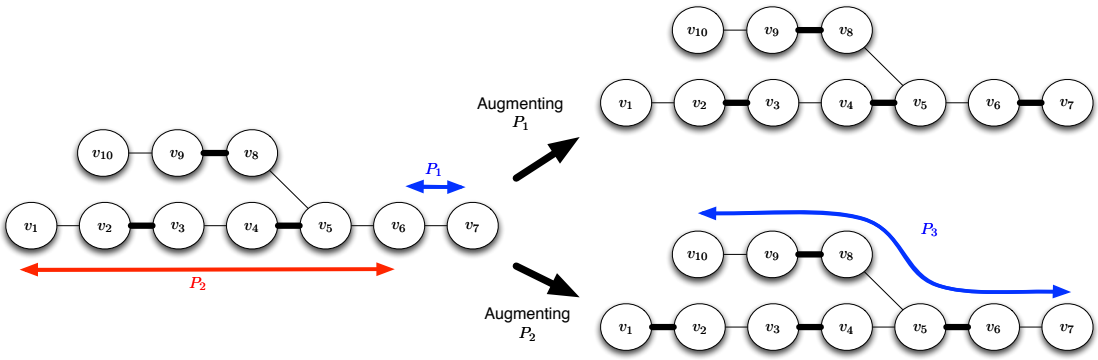



**Fig. 2.** Example of an instance (with $k = 5$) where augmenting a path (Path $P_2$) creates a new augmenting path ($P_3$) that must be augmented to reach an optimal solution. The matchings are represented in bold edges.

The cases $k \geq 3$ are more challenging because the order in which the paths are augmented is important. This fact is illustrated in Figure 1 where augmenting first the path ($A2C3$) leads to a non-optimal dead end configuration. In addition, the order in which the paths are augmented has an impact on the creation or non-creation of new augmenting paths of length at most $k$. For instance, for $k = 5$, let us consider the graph that consists of a path $(v_1, \cdots, v_7)$ plus three edges $\{v_5, v_8\}, \{v_8, v_9\}$, and $\{v_9, v_{10}\}$. The initial matching is $\{\{v_2, v_3\}, \{v_4, v_5\}, \{v_8, v_9\}\}$. This example is depicted in Figure 2. Initially, there are 2 augmenting paths of length at most 5: $P_1 = (v_6, v_7)$ and $P_2 = (v_1, \cdots, v_6)$. If $P_1$ is augmented first, then no augmenting paths of length at most 5 remain anymore. However, augmenting $P_2$ first leads to a configuration where path $P_1$ is not augmenting anymore but where a new path $P_3 = (v_{10}, v_9, v_8, v_5, v_6, v_7)$ can be augmented.

The main difference between the cases $k = 3$ and $k \geq 5$ is that, in the former case, it is possible to ignore the augmenting paths that were not present in the initial matching $M$. In other words, when $k = 3$, if a new augmenting path $P$ of length at most $k$ is created after augmenting some path, then it is not necessary to augment $P$ in order to achieve a matching of maximum size $\mu_3(G, M)$. The case when $G$ is restricted to be a path has similar properties for every odd $k \geq 1$. The next section is dedicated to prove these facts and to use them to design polynomial-time algorithms that compute $\mu_3(G, M)$ in general graphs and $\mu_k(P, M)$ in the class of paths, for every odd $k \geq 1$.

## 4 $k$-Matching Problem: polynomial cases

In this section, we present two polynomial-time algorithms. The first one (Subsection 4.1) computes $\mu_3(G, M)$ and a corresponding matching for any graph $G$, starting from any matching $M$. The second one (Subsection 4.2) computes $\mu_k(P, M)$ and a corresponding matching for any odd $k \geq 1$ and any path $P$ starting from any matching $M$. Both algorithms rely on the fact that, in both cases, only augmented paths that are initially present (i.e., $M$-augmenting paths) have to be considered.

### 4.1 Case $k = 3$ in general graphs

Let $G$ be a graph and $M \subseteq E(G)$ be a matching of $G$, and let $\mathcal{P}_3(G, M)$ denote the set of $M$-augmenting paths of length at most 3 in $G$.

**Lemma 1.** *Let $G$ be a graph and $M$ be a matching in $G$. Then, there exists $\mathcal{P} \subseteq \mathcal{P}_3(G, M)$ such that a matching of size $\mu_3(G, M)$ can be obtained from $M$ by augmenting the paths in $\mathcal{P}$ in any order.*

*Proof.* Let $M = M_0$. Let $(P_0, \cdots, P_r)$ be a sequence of paths of length at most 3 such that, for any $0 \leq i \leq r$, $M_{i+1} = M_i \Delta E(P_i)$ and $P_i \in \mathcal{P}_3(G, M_i)$ and such that $|M_{r+1}| = \mu_3(G, M)$. That is, $(P_0, \cdots, P_r)$ is an optimal solution of the 3-matching problem in $G$ starting from $M$. Note that, $\mu_3(G, M) = |M| + r + 1$.

- Case 1: Assume that $P_i \in \mathcal{P}_3(G, M)$ for any $i \leq r$. In this case, we prove by contradiction that these paths are pairwise vertex-disjoint, that is, for any $0 \leq i < j \leq r$, $V(P_i) \cap V(P_j) = \emptyset$. Assume that there exists $v \in V(P_i) \cap V(P_j)$ for some $0 \leq i < j \leq r$. There are two cases to be considered.
  - First, if $v$ is an end of $P_j$, then it is initially exposed and so, $v$ is also an end of $P_i$. When $P_i$ is augmented, $v$ becomes covered and cannot be exposed again following Property 1. Which implies that $P_j$ is not an augmenting path when it is its turn to be augmented, a contradiction.
  - Second, let us assume that $v$ is not an end of $P_j$. It means that $v$ is initially covered by $M$. This implies that $P_i = (a, v, w, b)$ and $P_j = (c, v, w, d)$ have the same "central" edge (possibly, they also share one of their ends). When $P_i$ is augmented, the edge $\{a, v\}$ becomes part of the matching and the edge $\{v, w\}$ does not belong to it anymore. Since $\{a, v\}$ was not in $M$ and $v$ was covered by $M$, no paths in $P_{i+1}, \cdots, P_{j-1}$ have $\{a, v\}$ as a "central" edge. Therefore, augmenting the paths $P_{i+1}$ to $P_{j-1}$ keeps $\{a, v\}$ in the matching. Therefore, when $P_j$ has to be augmented, it is not an augmenting path anymore, a contradiction.

  In both cases, we proved that the paths are pairwise vertex-disjoint. Therefore, they can be augmented in any order (they do not interfere with each other) and the lemma is proved.
- Case 2: Let $h \leq r$ be the maximum integer such that $P_h \notin \mathcal{P}_3(G, M)$. If such a path exists, the following process returns a new sequence (with the same number of paths) that either has strictly less paths not in $\mathcal{P}_3(G, M)$, or strictly decreases the largest index of such a path. Moreover, all expected properties hold, i.e., the paths of the obtained sequence are of length at most 3 and they can be sequentially augmented to obtain the desired matching. Hence, we will show that iterating this process eventually achieves a sequence of $M$-augmenting paths (i.e., all paths are in $\mathcal{P}_3(G, M)$) that results in the same matching as the initial sequence.

  First, let us prove that $P_h$ has length 3. If $P_h$ had length one, the extremities of $P_h$, $u$ and $v$, would be exposed by the matching $M_h$. Following Property 1, $u$ and $v$ would be exposed by $M_0$ and then $P_h \in \mathcal{P}_3(G, M)$, a contradiction.

  Hence, $P_h = (u, x, y, v)$, with $\{x, y\} \in M_h$ and $u, v$ are exposed in $M_h$. Then, since a covered vertex cannot become exposed (Property 1), $u, v$ are exposed in $M = M_0$. Moreover, $\{x, y\} \notin M_0$ because by assumption $P_h \notin \mathcal{P}_3(G, M)$. Let $0 \leq t < h$ be the maximum integer such that $\{x, y\} \notin M_t$. By definition of $t$, there is an $M_t$-augmenting path $P_t$ that is augmented to include $\{x, y\}$ in the matching. There are two cases to be considered.
  - Let us first assume that $P_t = (x, y)$. This implies that $x$ and $y$ were exposed (so, by Property 1, they were also exposed in $M_0$). In that case, we replace the sequence of augmenting paths by

    $$(P_1, \cdots, P_{t-1}, P_{t+1}, \cdots, P_{h-1}, (u, x), (y, v), P_{h+1}, \cdots, P_r).$$

    That is, the paths $P_t$ and $P_h$ are removed (they are not augmented anymore) and, instead, we augment the two paths $(u, x)$ and $(y, v)$ that are in $\mathcal{P}_3(G, M)$. It can be checked that this is a valid sequence of augmenting paths and it results in the same matching as the initial sequence. Hence, the obtained sequence contains strictly less paths that are not in $\mathcal{P}_3(G, M)$.
  - Second, let us assume that the considered $M_t$-augmenting path is $P_t = (a, b, x, y)$. That is, $a$ and $y$ are exposed in $M_t$ (and so, by Property 1, in $M_0$) and $\{b, x\} \in M_t$. Moreover, $a$ and $b$ are covered in $M_{t+1}$ (and so, by Property 1, in $M_h$) and then $\{a, b\} \cap \{u, v\} = \emptyset$. This case is depicted in Figure 3. In that case, we replace the sequence of augmenting paths by

    $$(P_1, \cdots, P_{t-1}, (a, b, x, u), P_{t+1}, \cdots, P_{h-1}, (y, v), P_{h+1}, \cdots, P_r).$$

    That is, instead of augmenting $P_t$, the path $(a, b, x, u)$ is augmented, and instead of augmenting $P_h$, the path $(y, v) \in \mathcal{P}_3(G, M)$ is augmented. It can be checked that this is a valid sequence of augmenting paths and it results in the same matching as the initial sequence. Hence, the largest index of a path that is not in $\mathcal{P}_3(G, M)$ has been decreased.

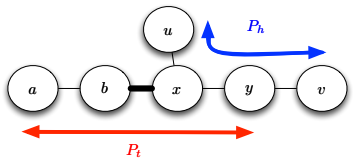



**Fig. 3.** Last case of proof of Lemma 1. The bold edge $\{b, x\}$ belongs to the matching $M_t$.

Hence, there exists $\mathcal{P} \subseteq \mathcal{P}_3(G, M)$ such that a matching of size $\mu_3(G, M)$ can be obtained from $M$ by augmenting the paths in $\mathcal{P}$. Moreover, all these paths are initially present and can all be augmented in any order, because they do not interfere with each other as the first item of this proof shows. □

Let $G$ be a graph and $M \subseteq E(G)$ be a matching of $G$. An edge $e \in E(G) \setminus M$ is said to be *useless* if $e$ is adjacent to two edges of $M$, i.e., if each endpoint of $e$ is incident to one edge of $M$. Intuitively, such an edge is called useless because it will never be part of the matching whatever be the sequence of augmenting paths of length at most 3 that are augmented. An edge $e \in M$ is said to be *forced* if at least one of its endpoints has degree one in $G$. Such an edge will never be part of any augmenting path and, therefore, it will belong to any $k$-maximum matching (with respect to $M$).

Let $\hat{G}$ be the graph obtained from $G$ and the matching $M$ by first removing all useless edges (but keeping their endpoints), and then by recursively removing the forced edges and their endpoints. Let $\hat{M}$ be the matching resulting from $M$ in $\hat{G}$, i.e., $\hat{M}$ equals $M$ minus the forced edges. Note that $(\hat{G}, \hat{M})$ has neither useless edges nor forced edges. For the sake of readability, we will identify the paths in $\hat{G}$ with the corresponding paths in $G$. Finally, let $f_M = |M| - |\hat{M}|$ denote the number of forced edges that have been removed to obtain $\hat{G}$ and $\hat{M}$.

**Lemma 2.** *Let $G$ be a graph and $M$ a matching. Let $\hat{G}$ and $\hat{M}$ be respectively the reduced graph and matching as defined before and $f_M$ be the number of forced edges. Then the following equalities hold: $\mathcal{P}_3(G, M) = \mathcal{P}_3(\hat{G}, \hat{M})$ and $\mu_3(G, M) = \mu_3(\hat{G}, \hat{M}) + f_M$.*

*Proof.* First, no useless edge of $(G, M)$ can belong to an augmenting path of length at most 3. Similarly, none of the forced edges that are removed to obtain $(\hat{G}, \hat{M})$ can belong to an augmenting path of length at most 3. Hence, $\mathcal{P}_3(G, M) \subseteq \mathcal{P}_3(\hat{G}, \hat{M})$. Reciprocally, removing useless or forced edges cannot create new augmenting paths of length at most 3. Hence, $\mathcal{P}_3(G, M) = \mathcal{P}_3(\hat{G}, \hat{M})$.

Then, the result follows from Lemma 1. □

**Lemma 3.** *Let $G = (V, E)$ be a graph and $M \subseteq E$ be a matching such that every edge in $E \setminus M$ has at least one exposed end, and every covered vertex has degree at least two (i.e., there are neither useless nor forced edges in $(G, M)$). Then:*

- *$\mu_3(G, M)$ equals the size $\mu(G)$ of a maximum matching in $G$.*
- *Moreover, a sequence of augmenting paths of length at most 3 that can be sequentially augmented starting from $M$ and achieving a matching of size $\mu_3(G, M)$ can be computed in polynomial time.*

*Proof.* Obviously, $\mu_3(G, M) \leq \mu(G)$.

Let $M^*$ be any maximum matching in $G = (V, E)$. Let us describe a sequence of augmenting paths of length at most 3 of $G$ (starting from the matching $M$) and achieving a matching of size $\mu(G) = |M^*|$. Hence, it proves that $\mu_3(G, M) = \mu(G)$.

For any $e \in M$, let $H_e$ be the set of edges that consists of $e$ and any edge adjacent to $e$, i.e., $H_e = \{f \in E \mid e \cap f \neq \emptyset\}$. Let $H = \bigcup_{e \in M} \{H_e\}$. The key point is that, because $(G, M)$ has no useless edges, the sets $H_e$ are pairwise disjoint. Moreover, any path in $\mathcal{P}_3(G, M)$ is either one edge of $E \setminus H$ or contains $e \in M$ and its 3 edges are in $H_e$.

First, let us slightly modify the maximum matching $M^*$ (while preserving its maximum size). Note that, for any $e \in M$, $1 \leq |H_e \cap M^*| \leq 2$. For $i \in \{1, 2\}$, let $J_i = \{e \in M \mid |H_e \cap M^*| = i\}$ and $I_i = \{H_e \cap M^* \mid e \in J_i\}$.

For any $e \in M$ such that $|H_e \cap M^*| = 1$ (i.e., $e \in J_1$), let us replace $H_e \cap M^*$ by $e$ in $M^*$. More formally, let $M' = (M^* \setminus I_1) \cup \bigcup_{e \in J_1} \{e\}$. Then, $M'$ is a matching of the same size as $M^*$, i.e., $M'$ is also a maximum matching of $G$.

Note that $M'$ can be partitioned into three sets: $M'_1 = M' \setminus H = M^* \setminus H$ (each edge of $M'_1$ is a path in $\mathcal{P}_3(G, M)$), $M'_2 = J_1 = M' \cap M$ and $M'_3 = M' \cap M^* \cap H = \{f_1^e, f_2^e \mid e \in J_2\}$. In $M'_3$, for every $e \in J_2$, $f_1^e$ and $f_2^e$ are two edges of $H_e \setminus \{e\}$ and the path $P_e$ consisting of $f_1^e, f_2^e$ and $e$ belongs to $\mathcal{P}_3(G, M)$. Finally, for every $e \in J_2$ and $e' \in M'_1$, the path $P_e$ and the edge $e'$ are vertex-disjoint.

Let us now show that $M'$ can be obtained by augmenting paths in $\mathcal{P}_3(G, M)$. Indeed, starting from $M$, it is sufficient to augment the paths $P_e$ for every $e \in J_2$ and every single edge $e' \in M'_1$. The resulting matching is precisely $M'$.

The second part of the lemma directly follows from the fact that a maximum matching $M^*$ can be computed in polynomial time [Edm65]. Then, the operations described in this proof can clearly be performed in polynomial time. □

From Lemmas 2 and 3, we get:

**Theorem 1.** *$\mu_3(G, M)$ and a corresponding matching can be computed in polynomial time for any graph $G$ and matching $M$.*

*Proof.* The algorithm proceeds as follows. Compute $\hat{G}$ and $\hat{M}$ and apply any polynomial-time maximum matching algorithm on $\hat{G}$. By Lemma 2, an optimal solution for the 3-matching problem in $G$ starting from $M$ consists of the forced edges, i.e., the edges in $M \setminus \hat{M}$, together with the ones of an optimal matching in $\hat{G}$. By Lemma 3, the latter edges can be obtained by augmenting paths in $\mathcal{P}_3(G, M)$. □

In the remainder of this paper, the algorithm presented in Theorem 1 is referred to as *Optimal* 3*-matching*. It is important to note that *Optimal* 3*-matching* precisely solves the recovery of disrupted airline operations presented in Section 1. In the last section (Section 6), we present results of simulations and analyse the behaviour of *Optimal* 3*-matching* on real-world instances.

### 4.2 Case of paths for any odd $k$

In this section, we consider the case of paths. We prove that, for any odd $k \geq 1$ and any matching $M$ of a path $P$, the algorithm given below returns, in linear-time, a sequence of augmenting paths of length at most $k$ (starting from $M$) and achieving a matching of size $\mu_k(P, M)$.

Intuitively, in a path, augmenting greedily the augmenting paths may not be optimal. For instance, consider the case $k = 3$ and the path $(v_0, \cdots, v_9)$ such that $M = \{\{v_1, v_2\}, \{v_4, v_5\}, \{v_7, v_8\}\}$. If we first augment the path $(v_3, v_4, v_5, v_6)$, we reach a situation where no augmenting path of length at most three exists. On the other hand, if we augment the paths starting from an extremity of the path graph we reach the optimal matching (e.g. first $(v_0, v_1, v_2, v_3)$ and then $(v_6, v_7, v_8, v_9)$). The main result of this section is that augmenting the paths of length at most $k$ starting from an extremity of a path graph will always reach an optimal solution.

More formally, the algorithm proceeds as follows. It starts from an end $v_0$ of the path. Let $Q = (v_0, \cdots, v_i)$ be the maximum alternating path containing $v_0$. If $Q$ is not augmenting or if $Q$ has length more than $k$ (i.e., $i > k$), the algorithm is applied recursively on $P \setminus (v_0, \cdots, v_{i-1})$. Otherwise, $Q$ is augmented (it is added to the desired sequence of paths) and the algorithm is applied recursively on $P \setminus Q$.

**Theorem 2.** *For any odd integer $k$, path $P_n = (v_0, \cdots, v_n)$, and matching $M \subseteq E(P_n)$, Algorithm $k$-matching$(P_n, M)$ computes, in linear-time, a sequence of augmenting paths of length at most $k$ that allows to reach a matching of size $\mu_k(P, M)$ starting from $M$.*

**Algorithm 1** k-matching$(P_n, M)$

**Require:** A path $P_n = (v_0, \cdots, v_n)$, a matching $M_1 = M \subseteq E(P)$, and an odd integer $k \geq 1$.
**Ensure:** A sequence $(Q_1, \cdots, Q_r)$ of paths, of length at most $k$ each, such that, for any $i \leq r$, $Q_i$ is $M_i$-augmenting and $M_{i+1} = M_i \Delta(Q_i)$, and $r = \mu_k(P, M) - |M|$ (i.e., $|M_{r+1}| = \mu_k(P, M)$).
1: Let $0 < i \leq n$ be the smallest integer such that $v_i$ is exposed.
2: **if** $i$ does not exist (*may occur if $M$ covers all vertices in $G$ but $v_0$*) **then**
3: **return** (), i.e., the empty sequence
4: **else if** $v_0$ is covered by $M$ OR $i > k$ **then**
5: Let $P' = (v_i, \cdots, v_n)$ and $M' = M \cap E(P')$.
6: **return** k-matching$(P', M')$
7: **else**
8: Let $P' = (v_{i+1}, \cdots, v_n)$ and $M' = M \cap E(P')$ (if $i = n$, then $P'$ is the empty path).
9: **return** $((v_0, \cdots, v_i)) \odot$k-matching$(P', M')$, i.e., the sequence consisting of the path $(v_0, \cdots, v_i)$, concatenated with the sequence k-matching$(P', M')$

*Proof.* The fact that the algorithm terminates in linear-time and that it returns a sequence of augmenting paths of length at most $k$ is obvious. It only remains to prove that this sequence allows to achieve a matching of size $\mu_k(P, M)$.

The proof is done by induction on $n$ and the result is obvious for $n \leq 0$ (Lines 2-3). Let us assume that $n > 0$.

- If $v_0$ is covered by $M$, or if it belongs to an $M$-augmenting path of length greater than or equal to $k+1$, then the only paths that may be augmented are the ones in $P' = (v_i, \cdots, v_n)$ where $i$ is the smallest integer at least 1 such that $v_i$ is exposed by $M$. Indeed, no edge of $M \cap E(v_0, \cdots, v_i)$ can ever be modified by any sequence of augmenting paths of length at most $k$ (by Property 1, all vertices in $\{v_0, \cdots, v_{i-1}\}$ that are initially covered will remain covered whatever be the paths that are augmented). Hence, the result follows by induction on the length of $P'$ (Line 6).
- Otherwise, let $Q = (v_0, \cdots, v_i)$ be the augmenting path containing $v_0$ such that $i \leq k$. We need to show that there exists a sequence of augmenting paths starting with $Q$ that allows to achieve a matching of size $\mu_k(P, M)$. Then, after having augmented $Q$, no edge of $M \cap E(v_0, \cdots, v_{i+1})$ can ever be modified by any sequence of augmenting paths (indeed, $v_0$ is a leaf and, once it becomes covered, no path containing $v_0, \cdots, v_i$ can be augmented). Hence, the result follows by induction on the length of $P'$ (Line 9).
  Let $M^*$ be a maximum matching obtained from $M$ by a sequence $\mathcal{S} = (Q_1, \cdots, Q_r)$ of augmenting paths of length at most $k$. Note that, by definition, $r = \mu_k(P_n, M) - |M|$. There are two cases to be considered.
  - If $\{v_0, v_1\} \in M^*$, because $v_0$ is exposed by $M$, there must exist $1 \leq j \leq r$ such that $Q_j$ is the augmenting path of $\mathcal{S}$ with endvertex $v_0$. Let $v_h$ be the other endvertex of $Q_j$ (Note that $h \geq i$ since, by Property 1 and by definition of $i$ (line 1 of Algorithm 1), no vertices in $\{v_1, \cdots, v_{i-1}\}$ can become exposed.) Note also that $Q_j$ is unique since once $\{v_0, v_1\}$ has been augmented it cannot be modified anymore because $v_0$ is a leaf. If $Q_j = Q$, we are done because $(Q_j = Q, Q_1, \cdots, Q_{j-1}, Q_{j+1}, \cdots, Q_r)$ is a valid sequence that leads to the same matching $M^*$. Otherwise, if $Q_j \neq Q$ (i.e., $h > i$), there is $\ell < j$ such that $Q_\ell = (v_i, \cdots, v_t)$ for some $i < t \leq h$. Let us set $Q'_j = (v_t, \cdots, v_h)$.
    It can be checked that the sequence $(Q, Q_1, \cdots, Q_{\ell-1}, Q_{\ell+1}, \cdots, Q_{j-1}, Q'_j, Q_{j+1}, \cdots, Q_r)$ is a sequence of augmenting paths of length at most $k$, achieving matching $M^*$.
  - Otherwise, there is $1 \leq j \leq r$ such that $v_i$ belongs to $Q_j$. Indeed, otherwise the sequence $(Q_1, \cdots, Q_r, Q)$ would lead to a larger matching. In that case, it can be checked that $(Q, Q_1, \cdots, Q_{j-1}, Q_{j+1}, \cdots, Q_r)$ leads to a matching of the same size than $M^*$ (i.e., we start by augmenting $Q$ and then, the path $Q_j$ is not augmented). □

The previous algorithm relies highly on the fact that, only augmenting paths that are initially present have to be augmented to achieve an optimal solution in paths (i.e., $M$-augmenting paths when starting from matching $M$). This is not the case anymore when the input graph is a tree (as shown in Figure 2). Hence the computational complexity of computing $\mu_k(T, M)$ in the class of trees $T$ is still an open problem.

## 5 NP-hardness when $k \geq 5$

In the previous section, we prove that the 3-matching Problem can be solved in polynomial time mainly because it is never required to augment augmenting paths that were not initially present. We also illustrate that, when $k \geq 5$ (even when the input graph is a tree), it may be the case that paths not initially augmenting have to be augmented to achieve an optimal solution. In this section, we prove that this makes the problem much harder to solve.

**Theorem 3.** *Let $k \geq 5$ be an odd integer. Computing $\mu_k(G, M)$ is NP-hard in the class of bipartite planar graphs $G$ with maximum degree 3, starting from any matching $M$.*

*Proof.* The proof is a reduction from planar exact-3-SAT, a well known NP-hard problem [Man83].

Let us consider an instance of planar exact-3-SAT, which is a formula $\Phi$, such that the corresponding bipartite graph $B_\Phi$ is planar. Formula $\Phi$ is composed of variables $v_1, \cdots, v_n$ and clauses $C_1, \cdots, C_m$ with exactly 3 literals each. $B_\Phi$ consists of the vertex set $\{v_1, \cdots, v_n, C_1, \cdots, C_m\}$ and for each $i$ and $j$ there is an edge $\{v_i, C_j\}$ only if Variable $v_i$ appears in Clause $C_j$. For any $i \leq n$, let $o_i$ denote the number of clauses where variable $v_i$ appears (positively or negatively).

For our purpose, it is important to fix a planar embedding of $B_\Phi$. In particular, recall that a given planar embedding of $B_\Phi$ defines a rotation system: for every variable vertex $v$ in $V(B_\Phi)$, an ordering of the edges around $v$ (i.e., for each vertex $v$, a permutation of the edges incident to $v$). For any $i \leq n$, let $J_i = \{j^i_1, \cdots, j^i_{o_i}\}$ be the set of integers $j$ such that Variable $v_i$ appears (positively or negatively) in Clause $C_j$, in the (cyclic) order given by the (fixed) planar embedding of $B_\Phi$ (starting from any clause). This ordering will only be needed to ensure that the graph in our reduction is planar.

We aim at building a graph $G_\Phi$ and an initial matching $M_\Phi$ such that $\mu_k(G_\Phi, M_\Phi) = |M_\Phi| + 4m$ if and only if $\Phi$ is satisfiable. In other words, starting from $M_\Phi$, it is possible to augment $4m$ paths of length at most $k$ if $\Phi$ is satisfiable, otherwise only strictly less paths can be augmented.

*Gadget-graph $G_{ij}$.* Let $i \leq n$ and let $j \leq m$ be such that the variable $v_i$ appears in Clause $C_j$ (positively or negatively). Let us build the gadget-graph $G_{ij}$ as follows. The graph $G_{ij}$ is a tree with $3k-1$ vertices $\{u^{ij}_1, \cdots, u^{ij}_{3k-1}\}$. It is obtained from the disjoint union of a path $(u^{ij}_1, \cdots, u^{ij}_{2k+1})$ and of a path $(u^{ij}_{2k+2}, \cdots, u^{ij}_{3k-1})$ by adding an edge between $u^{ij}_k$ and $u^{ij}_{2k+2}$ if $v_i$ appears positively in $C_j$, or by adding the edge $\{u^{ij}_{k+2}, u^{ij}_{2k+2}\}$ if $v_i$ appears negatively in $C_j$. The initial matching $M_\Phi$ restricted to the subgraph $G_{ij}$ is denoted by $M_{ij} = M_\Phi \cap E(G_{ij})$ and consists of the edges $\{u^{ij}_2, u^{ij}_3\}$, $\{u^{ij}_4, u^{ij}_5\}, \cdots, \{u^{ij}_{k-1}, u^{ij}_k\}$, $\{u^{ij}_{k+2}, u^{ij}_{k+3}\}$, $\{u^{ij}_{k+4}, u^{ij}_{k+5}\}, \cdots, \{u^{ij}_{2k-1}, u^{ij}_{2k}\}$, and $\{u^{ij}_{2k+2}, u^{ij}_{2k+3}\}$, $\{u^{ij}_{2k+4}, u^{ij}_{2k+5}\}, \cdots, \{u^{ij}_{3k-3}, u^{ij}_{3k-2}\}$.

Finally, for ease of presentation, let us denote $u^{ij}_1$ by $t_{ij}$, $u^{ij}_{2k+1}$ by $f_{ij}$, $u^{ij}_{k+1}$ by $c_{ij}$, $u^{ij}_{3k-1}$ by $v_{ij}$, $u^{ij}_k$ by $p_{ij}$, and $u^{ij}_{k+2}$ by $n_{ij}$.

Note that in $G_{ij}$, there are exactly two $M_{ij}$-augmenting paths of length at most $k$: the path $T_{ij}$ from $c_{ij}$ to $t_{ij}$ and the path $F_{ij}$ from $c_{ij}$ to $f_{ij}$ (Note that this is true because $k \geq 5$. Indeed, for $k = 3$, there would be another augmenting path of length $k$, namely $(t_{ij}, u^{ij}_{k-1}, p_{ij}, v_{ij})$ if Variable $v_i$ appears positively in $C_j$ or $(f_{ij}, u^{ij}_{k+3}, n_{ij}, v_{ij})$ otherwise.) Intuitively, augmenting the path $T_{ij}$ will correspond to assign $v_i$ to $True$ while augmenting the path $F_{ij}$ will correspond to assign $v_i$ to $False$.

The gadget-graph $G_{ij}$ and the initial matching $M_{ij}$ are depicted in Figure 4.

*Variable-graph $G_i$.* Let $i \leq n$ and let us build a Variable-gadget graph that corresponds to Variable $v_i$. Recall that $J_i = \{j^i_1, \cdots, j^i_{o_i}\}$ denotes the set of integers $j$ such that Variable $v_i$ appears (positively or negatively) in Clause $C_j$ in an order consistent with a fixed planar embedding of the bipartite graph $B_\Phi$.

The Variable-graph $G_i$ is obtained from the disjoint union of the gadget-graphs $G_{i,j^i_1}, \cdots, G_{i,j^i_{o_i}}$ by combining them into a "cycle", i.e., the paths between $t_{i,j^i_\ell}$ and $f_{i,j^i_\ell}$ in $G_{i,j^i_\ell}$ are "glued together" for $\ell = 1$ to $\ell = o_i$. More precisely, let us identify $t_{i,j^i_1}$ and $f_{i,j^i_{o_i}}$ and, for $1 \leq \ell < o_i$, let us identify $t_{i,j^i_{\ell+1}}$ and $f_{i,j^i_\ell}$.

The initial matching $M_i$ in $G_i$ is simply the union of the initial matchings $M_{ij}$ for $j \in J_i$, i.e., $M_i = \bigcup_{j \in J_i} M_{ij}$. It is important to note that the augmenting paths in $G_i$ are the augmenting paths in the $G_{ij}$'s,

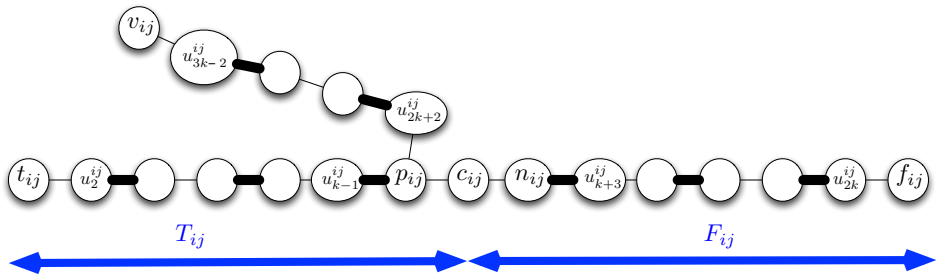



**Fig. 4.** Gadget-graph $G_{ij}$ for Variable $v_i$ appearing positively in Clause $C_j$. Edges of the initial matching $M_{ij}$ are depicted in bold. If Variable $v_i$ appears negatively in Clause $C_j$, then the edge $\{p_{ij}, u^{ij}_{2k+2}\}$ is replaced by the edge $\{n_{ij}, u^{ij}_{2k+2}\}$.

for $j \in J_i$, i.e., no new augmenting paths are created by combining the $G_{ij}$'s. This follows from the fact that the $G_{ij}$'s are "joined" by identifying exposed vertices, hence each augmenting path is fully included in some $G_{ij}$. An example is depicted in Figure 5.

Note that the order in which the graphs $G_{ij}$ are combined will be relevant only for ensuring the planarity of the graph. However, keeping the same "orientation" for each of them (i.e., the end $t$ of one such graph being identified to the end $f$ of the next one) is crucial. The key point of combining the graphs $G_{ij}$ in such a way is that the only way to augment one of the paths $T_{ij}$ or $F_{ij}$ for all $j \in J_i$ is to make the same choice for each $j \in J_i$, i.e., either all paths $T_{ij}$, $j \in J_i$, are augmented, or all paths $F_{ij}$, $j \in J_i$, are augmented. In other words, if there is $x, y \in J_i$ such that $T_{ix}$ and $F_{iy}$ are augmented, then there exists $z \in J_i$ such that neither $T_{iz}$ nor $F_{iz}$ can be augmented.

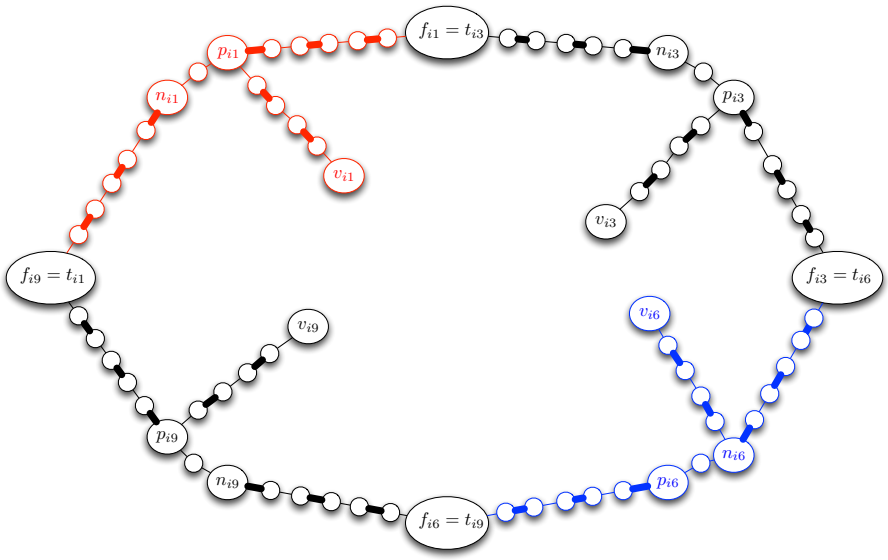



**Fig. 5.** Variable-gadget $G_i$ for $k = 7$ and Variable $v_i$ appearing positively in Clauses $C_1, C_3, C_9$ and negatively in Clause $C_6$. The edges of the initial matching appear in bold. Colors are only used to better distinguish the four gadget-graphs $G_{i1}$ (red), $G_{i3}$ (black), $G_{i6}$ (blue), and $G_{i9}$ (black).

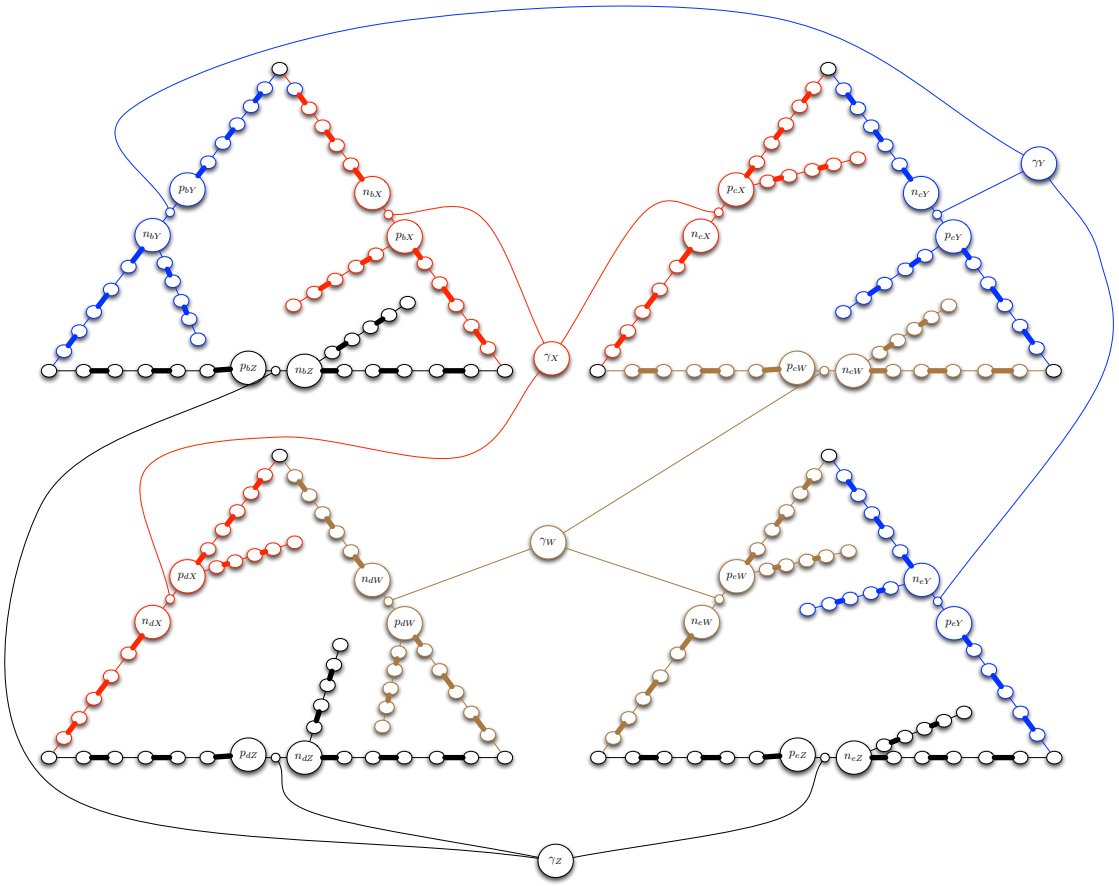



**Fig. 6.** Graph $G_\Phi$ and matching $M_\Phi$ built for $k = 7$ and from the formula $\Phi = (b \vee c \vee d) \wedge (\bar{b} \vee c \vee \bar{e}) \wedge (\bar{c} \vee d \vee e) \wedge (\bar{b} \vee \bar{d} \vee \bar{e}) = X \wedge Y \wedge Z \wedge W$ with Variables $b, c, d, e$ and Clauses $X = b \vee c \vee d$ (red) , $Y = \bar{b} \vee c \vee \bar{e}$ (blue), $W = \bar{c} \vee d \vee e$ (brown), and $Z = \bar{b} \vee \bar{d} \vee \bar{e}$ (black). The edges of the initial matching appear in bold.

*Main graph $G_\Phi$ and initial matching $M_\Phi$.* The main graph $G_\Phi$ is obtained from the disjoint union of the graphs $G_i$, $i \leq n$, and one vertex $\gamma_j$ for each Clause $C_j$, $j \leq m$, by adding an edge $\{\gamma_j, c_{ij}\}$ for any Variable $v_i$ appearing (positively or negatively) in Clause $C_j$. Finally, the initial matching $M_\Phi$ equals $\bigcup_{i \leq n} M_i$, i.e., the union of the initial matchings in the Variable-gadgets.

An example with formula $\Phi = (b \vee c \vee d) \wedge (\bar{b} \vee c \vee \bar{e}) \wedge (\bar{c} \vee d \vee e) \wedge (\bar{b} \vee \bar{d} \vee \bar{e})$ is depicted in Figure 6.

*Claim.* $G_\Phi$ is bipartite planar and with maximum degree 3.

*Proof.* It is bipartite with vertex-partition $(\alpha, \beta)$ where $\alpha$ consists of the vertices $u_h^{ij}$ with $h$ even, and $\beta$ consists of the vertices $\gamma_j$, $j \leq m$, union the vertices $u_h^{ij}$ with $h$ odd.

For any Variable $v_i$ appearing in Clause $C_j$, the vertex $v_{ij}$ has degree one. The vertices $\gamma_j$, $j \leq m$, and, for any Variable $v_i$ appearing in Clause $C_j$, the vertex $c_{ij}$ and one of the vertices $p_{ij}$ or $n_{ij}$, have degree three. All other vertices have degree two.

The planarity deeply relies on the ordering of the sets $J_i$. A planar embedding of $G_\Phi$ can be obtained as follows. First, start with a planar embedding of the bipartite graph $B_\Phi$. Then, for any $i \leq n$, the variable-vertex $v_i$ of $B_\Phi$ is replaced by the Variable-graph $G_i$. Recall that $G_i$ consists of a cycle plus $o_i$ paths attached to it (i.e., defining two faces, the bounded one and the unbounded one). To ensure the planarity, the paths

are embedded in the bounded face of $G_i$ while the Clause vertices are in the unbounded face. Clearly, a corresponding embedding is planar because $J_i$ respects the ordering of the planar embedding of $B_\Phi$. ◇

Finally, we need some additional notation. For any $i \leq n$, $j \leq m$ such that Variable $v_i$ appears in Clause $C_j$, let $R_{ij}$ be the shortest path in $G_\Phi$ between $\gamma_j$ and $v_{ij}$ and let $S_{ij}$ be the shortest path in $G_\Phi$ between $\gamma_j$ and $c_{ij}$. Note that $S_{ij}$ is initially an augmenting path of length 1. Intuitively, augmenting the path $T_{ij}$ corresponding to Variable $v_i$ appearing positively in Clause $C_j$ will allow to augment the path $R_{ij}$. Conversely, augmenting the path $F_{ij}$ will allow to augment the path $R_{ij}$ if the corresponding Variable $v_i$ appears negatively in Clause $C_j$.

*Claim.* If $\Phi$ is satisfiable, then $\mu_k(G_\Phi, M_\Phi) \geq |M_\Phi| + 4m$

*Proof.* Consider a truth assignment for $\Phi$.

For any $i \leq n$, if $v_i$ is assigned to $True$, then augment the path $T_{ij}$ for every $j \in J_i$. If $v_i$ is assigned to $False$, then augment the path $F_{ij}$ for every $j \in J_i$. In that way, for each $i \leq n$, $o_i$ paths are augmented. Since $\sum_{i \leq n} o_i = 3m$ (since each clause contains 3 variables), it makes a total of $3m$ paths of length at most $k$ that are augmented (independently).

Finally, for any $j \leq m$, let $i_j$ be one variable such that $v_{i_j}$ validates $C_j$. Then augment the path $R_{i_j,j}$.

In total, $4m$ paths are augmented and the resulting matching then has size $|M_\Phi| + 4m$. ◇

*Claim.* If $\mu_k(G_\Phi, M_\Phi) \geq |M_\Phi| + 4m$, then $\Phi$ is satisfiable.

*Proof.* We show that any sequence $\mathcal{S}$ of augmenting paths of length at most $k$ allowing to achieve some matching $M^*$ of size at least $|M_\Phi| + 4m$ from $M_\Phi$ has the following properties. Precisely, $\mathcal{S}$ consists of the following: for each $i \leq n$ and for $j \in J_i$, exactly one of the following cases holds. Case 1: all augmenting paths $T_{ij}$ belong to $\mathcal{S}$, in which case Variable $v_i$ is assigned to True. Case 2: all augmenting paths $F_{ij}$ belong to $\mathcal{S}$, in which case Variable $v_i$ is assigned to False. Then, for every $j \leq m$, there is one path $R_{ij}$, $j \in J_i$ that belongs to $\mathcal{S}$. In which case Variable $v_i$ satisfies Clause $C_j$. Hence $\mathcal{S}$ defines a truth assignment for $\Phi$.

Let us consider the behaviour of the combination of a gadget-graph $G_{ij}$ with Clause-vertex $\gamma_j$ where Variable $v_i$ appears positively in Clause $C_j$ (the case when $v_i$ appears negatively in $C_j$ is similar). In the subgraph induced by $G_{ij}$ and $\gamma_j$, there are initially three augmenting paths of length at most $k$, $T_{ij}$, $F_{ij}$, and $S_{ij}$. See Figure 7. That is a total of $9m$ paths that can be initially augmented.

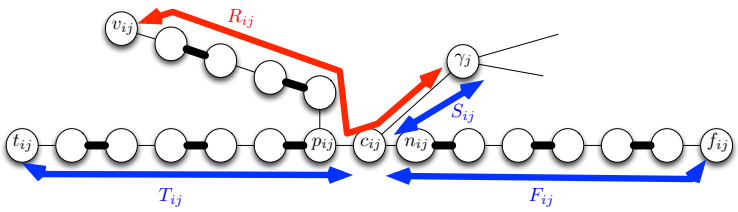



**Fig. 7.** Initial augmenting paths $T_{ij}$, $F_{ij}$, and $S_{ij}$ in the subgraph induced by $G_{ij}$ and $H_j$ (case when Variable $v_i$ appears positively in Clause $C_j$).

Augmenting the path $S_{ij}$ is depicted in Figure 8. Doing so does not create new augmenting paths of length at most $k$ and removes $T_{ij}$, $F_{ij}$, and $S_{i'j}$, for any Variable $v_{i'}$ in Clause $C_j$, from the possible augmenting paths.

Augmenting the path $F_{ij}$ is depicted in Figure 9. Doing so does not create new augmenting paths of length at most $k$ and removes $T_{ij}$ and $S_{ij}$ from the possible augmenting paths. It also removes $T_{ij'}$ from the

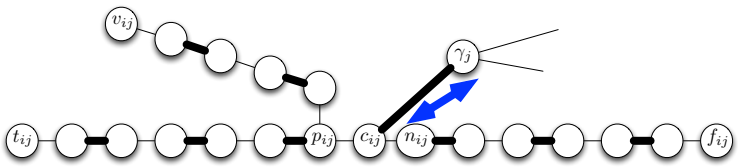



**Fig. 8.** Augmenting the path $S_{ij}$. (Case when Variable $v_i$ appears positively in Clause $C_j$.)

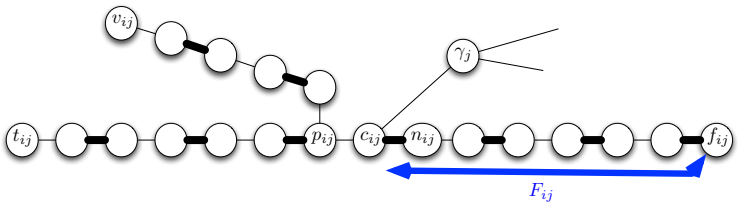



**Fig. 9.** Augmenting the path $F_{ij}$. (Case when Variable $v_i$ appears positively in Clause $C_j$.)

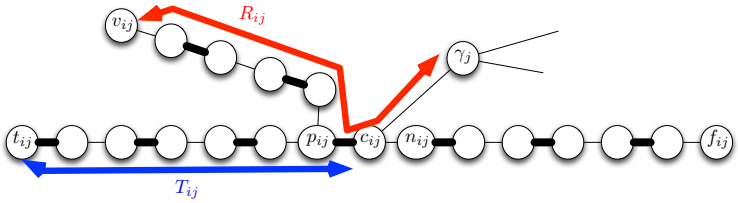



**Fig. 10.** Augmenting the path $T_{ij}$. (Case when Variable $v_i$ appears positively in Clause $C_j$.)

possible augmenting paths of length at most $k$ where $j'$ is the successor of $j$ in $J_i$ (recall that the elements of $J_i$ are ordered in a cyclic way).

Augmenting the path $T_{ij}$ is depicted in Figure 10. It does create the new augmenting path $R_{ij}$ of length at most $k$ and removes $F_{ij}$ and $S_{ij}$ from the possible augmenting paths. It also removes $F_{ij'}$ from the possible augmenting paths of length at most $k$ where $j'$ is the predecessor of $j$ in $J_i$ (following the cyclic order of the elements in $J_i$).

Finally, augmenting a path $R_{ij}$ does not create any new augmenting paths and causes that paths $S_{aj}$ and $S_{bj}$ cannot be augmented anymore, where $a$ and $b$ denote the indices of the other variables (different from $v_i$) appearing in $C_j$.

Recall that $o_i$ is the number of clauses in which $v_i$ appears. For any variable $v_i$, at most $o_i$ paths among $\{T_{ij}, F_{ij} \mid j \in J_i\}$ can be augmented. And in particular, there are $o_i$ such paths only if these paths are either all the paths in $\{T_{ij} \mid j \in J_i\}$, or all the paths in $\{F_{ij} \mid j \in J_i\}$. Moreover, since these $o_i$ paths have been augmented, none of the paths $\{S_{ij} \mid j \in J_i\}$ could have been augmented.

Consider any sequence $\mathcal{S}$ of augmenting paths allowing to achieve a matching $M^*$ of size $\geq |M_\Phi| + 4m$ from $M_\Phi$. By previous remarks, the number of paths in $\mathcal{S}$ containing some vertex in $\{\gamma_j \mid j \leq m\}$ is at most $m$. Hence, at least $3m$ paths must be augmented in $\bigcup_{i \leq n} \{T_{ij}, F_{ij} \mid j \in J_i\}$. Since $\sum_{i \leq n} o_i = 3m$, for any $i \leq n$, exactly $o_i$ paths must be augmented in $\{T_{ij}, F_{ij} \mid j \in J_i\}$. By the previous remark, it implies that, for every $i \leq n$, $\mathcal{S}$ contains either $\{T_{ij} \mid j \in J_i\}$, or $\{F_{ij} \mid j \in J_i\}$. This corresponds to our truth assignment: in the former case, $v_i$ is assigned to true, and to false in the latter case.

Moreover, by the previous remark, $\mathcal{S}$ contains no path in $\{S_{ij} \mid j \in J_i\}$. Since, for any $j \leq m$, $\mathcal{S}$ must contain a path that contains $\gamma_j$, there must be a path $R_{ij} \in \mathcal{S}$, for some $i \leq n$ with variable $v_i$ appearing in $C_j$. If $v_i$ appears positively (resp., negatively) in $C_j$, by construction, $R_{ij}$ can be augmented only if $T_{ij}$ (resp., $F_{ij}$) has been augmented. That is, $R_{ij}$ belongs to $\mathcal{S}$ implies that our boolean assignment satisfies $C_j$. Hence, if $|M^*| \geq |M_\Phi| + 4m$ from $M_\Phi$, the formula $\Phi$ can be satisfied. $\diamond$

## 6 Simulations

In the context of optimizing slot assignments, the airlines are competing against each other to capture the best slots as quickly as possible. Subsection 4.1 addresses partly the needs of airlines by presenting *Optimal* 3-*matching* to solve this problem (for computing $\mu_3(G, M)$ for any graph $G$ and initial matching $M$). In this section, we present results of simulations of *Optimal* 3-*matching* on real-world instances in order to assess the speed of the algorithm. To do so, we compare the running time of *Optimal* 3-*matching* with a heuristic and also with another algorithm solving the classical matching problem.

We have implemented the algorithm presented in Section 4.1 (for the case $k = 3$, see Theorem 1) using the graph library *SageMaths* (`www.sagemath.org`) in Python. Using this library has several advantages: it is well known and used in the graph community, it is interfaced with several solvers for linear programming, and many graph algorithms are already efficiently implemented. In particular, we have used the function `matching()` for computing "classical" maximum matchings (this function is based on a linear program solved using a solver).

*Greedy Algorithm.* We have also implemented a greedy algorithm for computing a matching augmenting only paths of length at most 3. This greedy algorithm takes a graph and an initial matching as inputs and performs as follows. At every iteration, it looks at the edges in a random order. For every edge $e = \{x, y\}$, the algorithm considers two cases. Case 1: none of the endpoints of $e$ are covered, then $e$ is added to the matching ($e$ is an augmenting path of length 1). Case 2: $e$ already belongs to the matching, the neighbours of $x$ and $y$ are checked in random order: if one uncovered neighbour $a$ of $x$ and one uncovered neighbour $b \neq a$ of $y$ are discovered, then the path $(a, x, y, b)$ is augmented. Each time a new path is augmented, the order of the edges of the graph is shuffled and a new iteration starts. The algorithm stops when no augmenting paths of length at most 3 exist anymore.

The code is available at: `http://www-sop.inria.fr/members/Nicolas.Nisse/kmatching_feb17.py` and all simulations have been done on a laptop with a 3,1 GHz Intel processor and 16Go of RAM.

*Instances.* We tested the algorithm on real-world instances. The instances are extracted from the Amadeus system and cover the free slots and slots assigned to the same airline at the airports of Brisbane, Melbourne, and Sydney, between July 2nd and July 23rd, 2015. Each of these 161 instances consists of a list of aircraft, the set of all the slots (those that are initially assigned and the free ones), and the initial assignment of the slots to the aircraft. The number of planes is between 70 and 271 with an average of 181. The total number of slots is between 115 and 1024 with an average of 554. Finally, the size of the initial matching (i.e., the number of slots that are initially assigned to aircraft) is between 15 and 212 with an average of 120. The distribution of the instances is depicted in Figure 11 by non-decreasing order of the number of aircraft.

*Analysis of the simulations.* For each of the 161 instances, we have run the proposed algorithm, the method `matching()` of *SageMaths*, and the greedy algorithm described above. We have compared the size of the matching each of these methods achieved and their corresponding running time. Since the greedy algorithm has a main random part, we have run it 50 times for every instance and kept the average size of the computed matching and the average running time.

The three plots in Figures 12 compare the size of the matching computed by the three algorithms. The instances are ordered by non-decreasing size of a "classical" maximum matching. Of course, the result of the proposed algorithm is always at most the size of a maximum matching and at least the size of the matching computed by the greedy algorithm. It can be noticed that it does not seem that the size of the initial

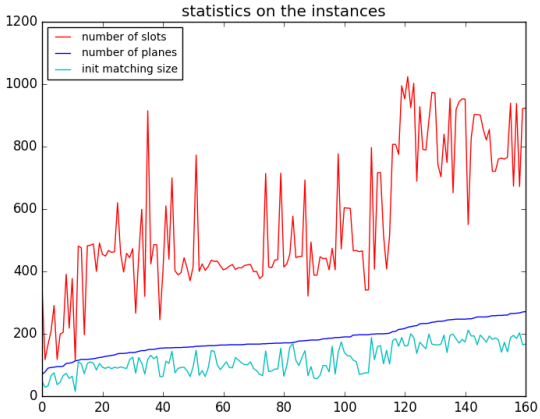



**Fig. 11.** Distribution of the number of aircraft, number of slots, and size of the initial matchings (on the $y$-axis, respectively in blue, red and cyan) of the 161 instances (on the $x$-axis). The instances are ordered by non-decreasing number of aircraft.

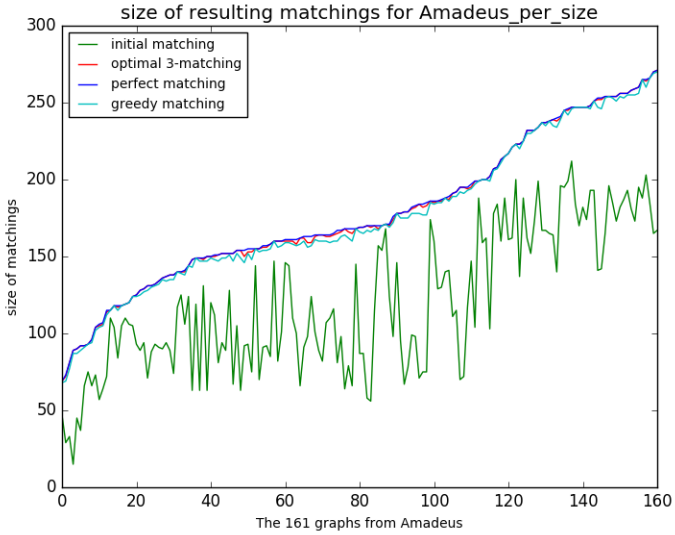
matching has any impact on the performance of the algorithms. Also, the results of the three algorithms are very close. However, in average, Algorithm Optimal 3-matching computes a matching whose size exceeds by two units the size of the matching computed by the greedy algorithm. This means that using Algorithm Optimal 3-matching could avoid two more cancelled flights per day and per airport (compared with a natural greedy solution).

Figure 13 shows the running time of the three algorithms (instances are ordered in non-decreasing running time of the `matching()` method). The fact that Algorithm Optimal 3-matching is (almost always) faster than the method `matching()` can be explained first because they do not achieve the same objective, but also because Algorithm Optimal 3-matching starts by reducing the graph (i.e., it computes $\hat{G}$ and $\hat{M}$ in Theorem 1) before applying `matching()` on a much smaller graph. As expected, the greedy algorithm is faster than the two other algorithms. However, on average, Algorithm Optimal 3-matching takes only 55ms more than the greedy algorithm (average on the 50 runs). Moreover, the order of magnitude of the execution time of Algorithm Optimal 3-matching (74ms on average and always less than 0.25 second) must be put in contrast with the time currently required for Air Traffic Controllers (we observed that typically more than 10 minutes are required).

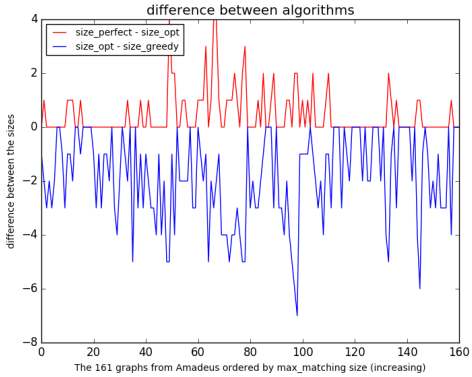
## 7 Conclusion and further work

This paper studies the computational complexity of the $k$-matching problem. In particular, we prove that the problem can be solved in polynomial time when $k \leq 3$. This is done by proving the optimality of a polynomial-time algorithm, that first consists in reducing the size of the instance and then applying an algorithm for computing a maximum matching (see the corresponding claim given in Section 3 for $k = 1$ and Theorem 1 for $k = 3$). In particular, for many practical instances (see Section 6), the first phase significantly reduces the size of the instance and, therefore, Algorithm Optimal 3-matching computes optimal solutions in a small amount of time. Consequently, we provide an efficient algorithm for solving the industrial problem motivating this paper: the optimal assignment of arrival slots to planes. In addition we also prove that the problem can be solved in polynomial time for any $k$ if the initial graph is a path.

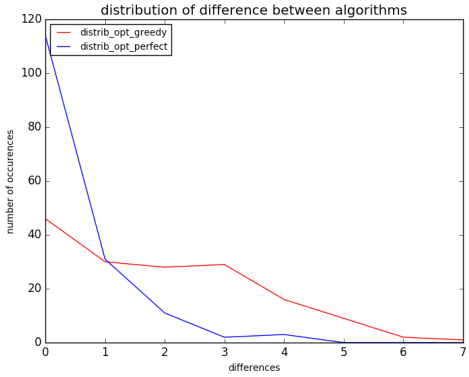
We show, however, that the problem is NP-hard when $k \geq 5$ in very specific graph classes, i.e. planar bipartite graphs with maximum degree 3. A natural open question is to investigate the case of other particular graph classes such as trees, or more generally, graphs with bounded treewidth.




(a) This plot indicates the size of the matching obtained by: Algorithm Optimal 3-matching (red), the greedy algorithm (light blue), and the method `matching()` (blue). The size of the initial matching is depicted in green.




(b) Difference between the three algorithms. In red: difference between a "classical" maximum matching and a maximum matching constrained with augmenting paths of length at most 3 (i.e., the result of Algorithm Optimal 3-matching). In blue: difference between the size of a matching obtained by the greedy algorithm (average over 50 executions) and the result of Algorithm Optimal 3-matching.




(c) Distribution of the differences. In red: at the abscissa $i$ is depicted the number of instances such that the difference between the size of the matching computed by Algorithm Optimal 3-matching and the one computed by the greedy algorithm equals $i$. In blue: at the abscissa $i$ is depicted the number of instances such that the difference between the size of a "classical" maximum matching and the size of the matching computed by Algorithm Optimal 3-matching equals $i$.

**Fig. 12.** Comparison of the size of the matchings obtained by the three algorithms. The instances are ordered by non-decreasing size of a "classical" maximum matching.

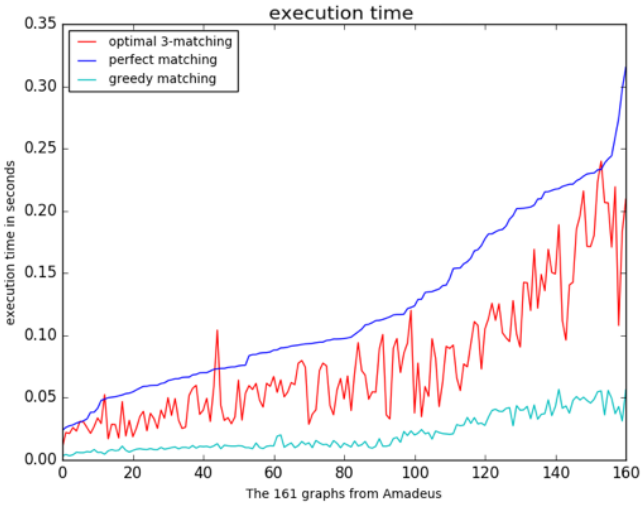



**Fig. 13.** Execution time of the three algorithms: the `matching()` method ("perfect matching"), Algorithm Optimal 3-matching, and the greedy algorithm. The instances are ordered in non-decreasing running time of the `matching()` method.

*Acknowledgement.* The authors would like to thank the anonymous reviewers for their time and efforts, and for providing valuable feedback in their reports.

## References

[Aus14] Airservices Australia. Metron traffic flow benefits realisation. `http://www.airservicesaustralia.com/wp-content/uploads/Metron-Traffic-Flow-Benefits-Realisation-Study-Further-benefits.pdf`, 2014. [Online; accessed 12-January-2017].

[Aus15a] Airservices Australia. Air traffic flow management business rules. `https://www.airservicesaustralia.com/noc/cdm/docs/CDM_ATFM_Business_Rules.pdf`, 2015. [Online; accessed 18-January-2017].

[Aus15b] Airservices Australia. Air traffic flow management user manual. `https://www.airservicesaustralia.com/noc/cdm/docs/ATFM_User_Manual_V9.pdf`, 2015. [Online; accessed 18-January-2017].

[Ber57] C. Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences of the United States of America*, 43(9):842–844, September 1957.

[BGN+17] Julien Bensmail, Valentin Garnero, Nicolas Nisse, Alexandre Salch, and Valentin Weber. Recovery of disrupted airline operations using k-maximum matching in graphs. In *9th Latin-American Algorithms, Graphs and Optimization Symp. (LAGOS)*. Elsevier, Electronic Note Discrete Maths, 2017.

[BSS09] L. Bui, S. Sanghavi, and R. Srikant. Distributed link scheduling with constant overhead. *IEEE/ACM Trans. Netw.*, 17(5):1467–1480, 2009.

[DP14] R. Duan and S. Pettie. Linear-time approximation for maximum weight matching. *J. ACM*, 61(1), 2014.

[Edm65] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[Eur12] Eurocontrol. Airport cdm implementation manual. `http://www.eurocontrol.int/sites/default/files/publication/files/2012-airport-cdm-manual-v4.pdf`, 2012. [Online; accessed 18-January-2017].

[HK73] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

[Kuh55] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[Man83] A. Mansfield. Determining the thickness of graphs is NP-hard. *Mathematical Proceedings of the Cambridge Philosophical Society*, 93(1):9–23, 1983.

[MV80] S. Micali and V. V. Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *21st Symp. on Foundations of Comp. Sc. (FOCS)*, pages 17–27. IEEE, 1980.

[Sie10] Gernot Sieg. Grandfather rights in the market for airport slots. *Transportation Research Part B: Methodological*, 44(1):29 – 37, 2010.

[Wam96] Michael Wambsganss. Collaborative decision making through dynamic information transfer. *Air Traffic Control Quarterly*, 4(2):109–125, 1996.

[WS05] X. Wu and R. Srikant. Regulated maximal matching: A distributed scheduling algorithm for multi-hop wireless networks with nodeexclusive spectrum sharing. In *IEEE Conf. on Decision and Control*, 2005.