Asynchronous Rendezvous with Different Maps

Serafino Cicerone Gabriele Di Stefano Leszek Gąsieniec Alfredo Navarra

Abstract

This paper provides a study on the rendezvous problem in which two anonymous mobile entities referred to as *robots* r_A and r_B are asked to meet at an arbitrary node of a graph $G = (V, E)$. As opposed to more standard assumptions robots may not be able to visit the entire graph G. Namely, each robot has its own *map* which is a connected subgraph of G. Such mobility restrictions may be dictated by the topological properties combined with the intrinsic characteristics of robots preventing them from visiting certain edges in E.

We consider four different variants of the rendezvous problem introduced in [Farrugia et al., *SOFSEM'15*] which reflect on restricted maneuverability and navigation ability of r_A and r_B in G. In the latter, the focus is on models in which robots' actions are synchronised. The authors prove that one of the maps must be a subgraph of the other. I.e., without this assumption (or some extra knowledge) the rendezvous problem does not have a feasible solution. In this paper, while we keep the containment assumption, we focus on asynchronous robots and the relevant bounds in the four considered variants. We provide some impossibility results and almost tight lower and upper bounds when the solutions are possible.

1 Introduction

The *Rendezvous* problem comprises the task of meeting two anonymous mobile robots which start at different nodes of a graph or different locations in the Euclidean space. Many variants (with different assumptions) of rendezvous have been studied in the past. An exhaustive survey on the problem can be found in [13], and some further advances in [2, 3, 4, 5, 9, 11, 14]. In this paper, we are interested in the design of deterministic algorithms for asynchronous robots moving across edges in the underlying graph of network connections. The deterministic and asynchronous variant of rendezvous in graphs has been first introduced in [7]. Later in [6], the problem has been fully characterised and the adopted model utilised the minimal setting under which the rendezvous can be accomplished. The authors of [6] give also the answer to the question posed in [7] whether there exists a deterministic algorithm for rendezvous of two asynchronous robots in any finite connected graph without knowing any upper bound on its size. The minimal assumptions to enable rendezvous include:

- The input anonymous graph has no labels on points. Instead, at each node of degree d, the relevant end points of incident edges are sorted and labelled by port numbers $1, \ldots, d$. The local labelling of ports at each node is fixed, i.e., every robot sees the same local labelling. However, no coherence between local labellings is assumed. I.e., one edge can have two different port numbers at its opposite ends. When a robot leaves a node, it is aware of the port number by which it leaves and when it enters a node, it is aware of the entry port number. It can also verify, at each node, whether a given positive integer is a port number at this node.
- Each robot has a unique ID, however, it does not know the ID of the other robot.
- Robots can meet on nodes or along edges, i.e., forcing robots to meet on nodes may prevent them from rendezvous.

In the model described above robots do not know G nor the initial distance between them in G. They cannot mark neither the nodes nor the edges. Rendezvous has to be accomplished for any local labelling of ports. The robots terminate their walks at the time of meeting one another. The rendezvous algorithm works also for infinite graphs. In fact, in finite graph the resolution of the rendezvous is often trivial or it can be reduced to the graph *exploration* problem. For example, utilising search methods proposed in [8] one can force to meet the two robots in finite tree. Namely, the rendezvous can be easily reached once both robots discover the centre(s) of the tree.

In this paper, however, we are interested in a different model in which robots have no IDs and most importantly they may not be allowed to access the whole graph. The roaming space of each robot is limited to a specific subsets of nodes and edges. The reasons to adopt such restriction may vary, however, the restriction itself is natural and was used earlier, e.g., in the evacuation problem [1] where an entity may represent a disabled person not able to adopt steep stairs or an escalator.

The rendezvous problem with heterogeneous (different accessibility restrictions) entities was formally introduced in [10] under the name of *rendezvous with different maps*. In the most general variant two asynchronous and anonymous robots r_A and r_B provided with two different maps G_A and G_B, both isomorphic to (possibly different) subgraphs G'_A and G'_B of the finite input graph G. The meeting can happen only on nodes but it is assumed that traversal of edges is mutually exclusive. This assumption is equivalent to the one used in [6], where robots can also meet on edges.

The main difference between the standard rendezvous problem studied, e.g., in [6] and the rendezvous with different maps studied here is in the way robots build their trajectories. In particular, in the latter the robots do not have to construct their maps (discover reachable nodes and edges). The maps are provided to them beforehand and the relevant trajectories can be precomputed prior to the actual search stage. This is in contrast to the model utilised in [6] where the trajectory of a robot is computed "on-the-go" on the basis of the current local information about port numbers, node degrees and the ID. Thus the main difficulty in the model adopted here refers to inconsistency of the maps provided to the robots in which one robot may not be able to access certain nodes or edges reachable for the other. Similar challenges occur also in blind rendezvous [12].

According to [10] without some extra information (e.g., node IDs) rendezvous with maps cannot be accomplished if $G'_A \nsubseteq G'_B$, where r_A is the robot with the smaller map. Thus here we also assume $G'_A \subseteq G'_B$. In contrast to [10], we focus solely on *asynchronous* robots. We study four natural variants of the rendezvous with different maps, combining two natural assumptions/properties considered in [10]: (1) availability of relative (with no explicit labels) ordering of nodes, and (2) presence of robot weights vs edge weight tolerance. The four variants are determined by the presence (or absence) of these two properties. We also discuss two hierarchies (that share the bottom and the top levels) formed by the four studied variants of the problem. At the top level of these hierarchies we assume presence of both properties. In the middle we have two incomparable levels where only one property is present. Finally at the bottom level we consider the absence of both properties.

We provide both the lower and the upper bounds with respect to the considered variants. We show that at the bottom level of the hierarchies very little can be done w.r.t. the rendezvous problem. In particular, the absence of the two properties makes the problem unsolvable in G with an arbitrary topology, and is tractable only in the case of simple topologies including paths and stars. We also show that in the two intermediate (and incomparable) variants rendezvous can be efficiently concluded in cycles and trees (the robots cannot rendezvous in cycles at the bottom level). Finally, we propose efficient (in terms of moves made) algorithm for the upper level requiring only $O(N \log N)$ steps. This result is almost tight in view of the natural lower bound of $\Omega(N)$, where N denotes the cumulative number of vertices of the two maps G'_A and G'_B.

Due to space limitations, figures are moved to Appendix.

2 Model

We start with a summary and further extension of the computation model introduced in [10]. We consider rendezvous of anonymous (and indistinguishable with respect to the control mechanism) robots in networks modelled by finite undirected graphs. The network $G = (V, E)$ is a simple connected graph, where $|V| = n$ and $|E| = m$. The two robots r_A and r_B initiate search at different *starting nodes* $s_A \neq s_B$ in G. Each robot $r_X \in \{r_A, r_B\}$ has its own *map* $G_X = (V_X, E_X)$ which is isomorphic to a specific subgraph $G'_X = (V'_X, E'_X)$ of G induced by the sets of nodes V'_X and edges E'_X *reachable* from s_X by robot r_X. In particular, the matching between the map of r_X and G'_X is deterministic and known to r_X. We emphasise that each robot r_X only knows its own map G_X and the starting node s_X. In other words r_A has no knowledge of G_B and s_B, and vice versa. Moreover, during search r_A cannot adopt edges outside of its map G_A and its trajectory is oblivious w.r.t. to the knowledge possessed by r_B. Note that, once r_X has computed its trajectory on its map, by the above assumptions, it can move on G'_X consistently, without ambiguities.

Let $n_X = |V_X|$ be the number of nodes of map G_X, $m_X = |E_X|$ be the number of edges of G_X, while by N and M we denote $n_A + n_B$ and $m_A + m_B$, respectively. Finally, given a node $v \in V$, the set of its neighbours is denoted by $N_G(v) = \{v' \mid (v, v') \in E\}$.

We assume that the robots act in asynchronous fashion. Each robot computes its trajectory, the sequence of visited nodes and edges, independently and prior to the actual search. We assume that the use of edges is exclusive, i.e., two robots cannot be located (move in either directions) on the same edge at any time. When the robot is ready to move along a chosen edge it awaits the relevant "green light" signal (meaning the edge is now available) from the system. In consequence, rendezvous is possible only on nodes when one robot is immobilised indefinitely or awaits access of an edge through which the other robot is approaching. The time required to move across an edge is assumed to be finite but unbounded. In turn, as the complexity of the solution we adopt the sum of the lengths of the robots' trajectories before rendezvous, i.e., the number of edges traversed in total.

In what follows we formalise four different variants of rendezvous with different maps. Each variant is determined by the availability of extra knowledge O and W (for the definition see below) w.r.t. the maps. For all considered variants, we assume that G'_A is a subgraph of G'_B. Otherwise, as already indicated, the rendezvous problem with different maps may not have a solution [10].

- Property O: the nodes of G are *totally ordered*. In particular, if $V = \{v_1, v_2, \ldots, v_n\}$ then $v_i < v_{i+1}$, for all $i = 1, 2, \ldots, n-1$. We say that Property O holds if this order is consistent with the order of nodes observed by robot r_X in G_X. That is, if $V_X = \{v_1^X, v_2^X, \ldots, v_{n_X}^X\}$, $v_p^X = v_i$, and $v_q^X = v_j$, where $v_i, v_j \in V$ and $i < j$, we also get $v_p^X < v_q^X$.
- Property W: each robot $r_X \in \{r_A, r_B\}$ has an associated weight $w_X \in \mathbb{R}^+$, and each edge $e \in E$ can tolerate weights up to the limit $w(e) \in \mathbb{R}^+$. In this setting let H_X denote the (possibly disconnected) subgraph of G induced by edges $e \in E$ such that $w(e) \geq w_X$. Then G'_X is the connected component of H_X which contains s_X. We assume that the maps $G_X = (V_X, E_X)$ contains information about the weights tolerated by the relevant edges (where $w(e^X) = w(e)$, for each $e^X \in E_X$ represents the edge $e \in E$).

We consider four variants based on properties O and W:

- WO **variant**, where both properties O and W hold,
- $\overline{\mathsf{W}}\mathsf{O}$ **variant**, where only O holds,
- $\mathsf{W}\overline{\mathsf{O}}$ **variant**, where only W holds,
- $\overline{\mathsf{WO}}$ **variant**, where neither O nor W holds.

By slightly abusing our notation the codes WO, $\overline{\mathsf{W}}\mathsf{O}$, $\mathsf{W}\overline{\mathsf{O}}$, and $\overline{\mathsf{WO}}$ will be used not only to define the variants of the problem, but also the set of instances of the relevant variants. For example, WO will refer to all instances of rendezvous with different maps where each robot r_X knows: (1) its *weighted* map G_X, (2) the starting point s_X, and (3) it is aware that its node ordering is consistent with the node ordering of the other robot. Using this notation one can define a relationship $\sqsubseteq$ between the elements of $\mathbb{V} = \{\mathsf{WO}, \overline{\mathsf{W}}\mathsf{O}, \mathsf{W}\overline{\mathsf{O}}, \overline{\mathsf{WO}}\}$. For example, $\overline{\mathsf{WO}} \sqsubseteq \overline{\mathsf{W}}\mathsf{O}$ means that for each instance $i \in \overline{\mathsf{WO}}$ it is possible to identify a set $I \subseteq \overline{\mathsf{W}}\mathsf{O}$ of *instances induced by* i as follows: if $i = (G_A, s_A, G_B, s_B)$, then each instance in I is obtained from i by maintaining (G_A, s_A, G_B, s_B) and by adding any possible consistent ordering on nodes of G_A and G_B. One can observe that such relationship defines two hierarchies: $\overline{\mathsf{WO}} \sqsubseteq \overline{\mathsf{W}}\mathsf{O} \sqsubseteq \mathsf{WO}$ and $\overline{\mathsf{WO}} \sqsubseteq \mathsf{W}\overline{\mathsf{O}} \sqsubseteq \mathsf{WO}$. The following holds.

Remark 1. *Let $\mathcal{V}_1, \mathcal{V}_2 \in \mathbb{V}$ such that $\mathcal{V}_1 \sqsubseteq \mathcal{V}_2$. If $i \in \mathcal{V}_1$ and $I \subseteq \mathcal{V}_2$ is the set of instances induced by i, then:*

- *if i is solvable in $\mathcal{V}_1$, then each induced instance in I is solvable in $\mathcal{V}_2$;*
- *if all the instances in I are unsolvable in $\mathcal{V}_2$, then i is unsolvable in $\mathcal{V}_1$.*

In the remaining part of the paper we propose and analyse algorithmic solutions for the rendezvous problem with different maps. Our algorithms assume each robot r_X has the input map G_X and the initial position s_X. The output of an algorithm refers the rendezvous trajectory computed by each r_X on G_X. The complexity of the solution is defined as the sum of the lengths of trajectories adopted by both robots until rendezvous takes place. For the sake of simplicity, knowing that G_X and G'_X are isomorphic and that r_X is aware of the isomorphism, in the following we always write G_X rather than G'_X even when we refer to the moves along edges in G'_X and the properties of G'_X.

3 Preliminary results

In this section we provide a general lower bound holding for all variants, and a more restrictive one which does not hold only for WO. Then we present a sufficient condition for solving the rendezvous problem that will be exploited successively by our resolution algorithms for variants WO and $\overline{\mathsf{W}}\mathsf{O}$. Finally, we provide optimal algorithms and infeasibility results for maps with specific topologies in the weaker variants $\overline{\mathsf{WO}}$ and $\mathsf{W}\overline{\mathsf{O}}$.

3.1 Lower bounds

The following lemma provides a lower bound on the length of the trajectory performed by robots in any solving algorithm with respect to the WO variant.

Lemma 1. *In variant* WO*, rendezvous requires use of trajectories of length* $\Omega(N)$.

Proof. Consider an instance of the problem where $n_A = 1$. Then, any rendezvous algorithm is stuck with r_A immobilised in the starting node s_A. Since r_B has no knowledge of the position of r_A, in the worst case it has to move throughout all the nodes of its map. □ □

Thus by Remark 1 and Lemma 1 the lower bound $\Omega(N)$ applies also in any variant in $\mathbb{V}$.

Lemma 2. *In variants* $\overline{\mathsf{W}}\mathsf{O}$ *and* $\mathsf{W}\overline{\mathsf{O}}$ *rendezvous requires use of trajectories of length* $\Omega(M)$.

Proof. Recall first that neither in variant $\overline{\mathsf{W}}\mathsf{O}$ nor in $\mathsf{W}\overline{\mathsf{O}}$ robots have enough information to meet in (agreed in advance) target node for their meeting.

In variant $\overline{\mathsf{W}}\mathsf{O}$ consider the case in which the map of r_B is formed of $m_B = \Omega(n_B^2)$ edges and the map of r_A is a single edge $e = \{v_1, v_2\}$. According to Lemma 1, any rendezvous algorithm $\mathcal{A}$ must force robots to visit all nodes in their map. Thus also r_A has to visit both nodes v_1 and v_2 by traversing the only edge at most once. If r_B traverses only $o(m_B)$ edges and stops, the adversary picks e among edges not traversed by r_B with the endpoints different to the node where r_B rests eventually. This is possible if the map of r_B is dense enough. During rendezvous, r_A is allowed first to access e and is kept there until r_B stops. Since the final node on r_B's trajectory is different to v_1 and v_2 rendezvous is not reached. In the complementary case, i.e., when r_B visits its whole map we assume that e is the last edge visited by r_B. Here also the adversary allows r_A to enter this edge first where r_A waits until r_B comes to visit this edge. This will force r_B to visit $\Omega(m_b) = \Omega(M)$ edges.

In variant $\mathsf{W}\overline{\mathsf{O}}$ consider a 3-layer graph $G = (V, E)$, where the set of nodes V is partitioned into three subsets V_1, V_2 and V_3 of the same size $\frac{n}{3}$. Also the set of edges is partitioned into E_1 and E_2, such that graphs $(V_1 \cup V_2, E_1)$ and $(V_2 \cup V_3, E_2)$ are complete bipartite graphs. We also assume that edge tolerance within each set E_i, for $i = 1, 2$, is uniform, however, edges in E_2 tolerate w_A but those in E_1 don't. In contrast, all edges in E tolerate w_B. Assume also that $s_A \in V_2$ and $s_B \in V_1$.

By Lemma 1, r_X cannot stay in s_X. Let e be the edge r_A traverses first. The adversary temporarily entraps r_A on e. If the trajectory computed by r_B is of length at least $\Omega(n^2)$, due to the uniform weight tolerance on edges in E_2 the adversary can pick e, s.t., occurs on r_B's trajectory only after $\Omega(n^2)$ steps. In the complementary case, when the trajectory of r_B is of length $o(n^2)$, the adversary picks e outside of the trajectory of r_B. In this case the adversary instructs r_B to move first entrapping it in the last edge e' of its trajectory. If the protocol for r_A is perpetual or of length $\Omega(n^2)$ due to uniformity of edges the adversary can force this protocol to walk $\Omega(n^2)$ edges before entering e', and the rendezvous takes place only if $e' \in E_2$. If the protocol for r_A is of length $o(n^2)$ the adversary keeps r_B away from e' and stops at its destination node v. Finally, r_V is released to finish walk at v'. As the robots cannot agree in advance to meet on the same target node, i.e., $v \neq v'$, there is no rendezvous in this case. □ □

It follows from Lemma 2 that in variants $\overline{\mathsf{W}}\mathsf{O}$ and $\mathsf{W}\overline{\mathsf{O}}$ (and by Remark 1 also in $\overline{\mathsf{WO}}$) any algorithm has to move robots through all edges of their respective maps. Whereas, in variant WO this is not true as robots could exploit knowledge about nodes' ordering and edges' weight tolerance.

3.2 A sufficient condition for solving rendezvous

In this section we provide a sufficient condition for solving rendezvous with different maps. We first formalise concepts of walks and sub-walks in a graph (cf Fig. 4). A *walk* in a graph G is an ordered sequence of edges of G, $W = ((v_{i_1}, v_{i_2}), (v_{i_2}, v_{i_3}), \ldots, (v_{i_{k-1}}, v_{i_k}))$, where the second node of an edge is the first node of the subsequent edge; in W, v_{i_1} is the starting node and v_{i_k} is the final node. By $|W|$, we denote the number of edges forming W. Given two walks W' and W'' in G, we write $W' \subseteq W''$ when W' is a *sub-walk* of W'', i.e., W' is a (not necessarily contiguous) sub-sequence of edges in W''. If a walk W contains all edges of G then it is called a *complete walk* of G.

Lemma 3. *Let* W_X *be a complete walk of map* G_X *which starts in* s_X*. If* $W_A \subseteq W_B$*, then the rendezvous is solvable even in variant* $\overline{\mathsf{W}}\mathsf{O}$.

Proof. An algorithm $\mathcal{A}$ can solve the rendezvous as follows: move robot r_X along walk W_X starting at s_X and finishing in the final node of W_X, unless the rendezvous is accomplished earlier. Since $W_A \subseteq W_B$, robot r_B has to visit all edges in W_A in the same order as robot r_A does. Thus no adversary can force r_B to overpass r_A on W_A despite actions of robots being asynchronous. □ □

3.3 On the complexity of the rendezvous problem in variants $\overline{\mathsf{WO}}$ and $\mathsf{W}\overline{\mathsf{O}}$

We start the discussion of rendezvous with different maps in variant $\overline{\mathsf{WO}}$.

As already discussed, one can find in [6] full characterisation of the standard asynchronous rendezvous problem, including the minimal assumptions under which the rendezvous can be accomplished. These include (1) consistent port numbering for the two maps, (2) unique IDs of robots, and (3) meeting allowed at nodes and edges. Consider now variant $\overline{\mathsf{WO}}$ variant with an instance in which $G_A = G_B$. In such case, one can claim that "rendezvous with different maps" is equivalent to "standard rendezvous problem" when neither port numbering nor node IDs are provided.

Thus using the argument above and [6] we get the following theorem.

Theorem 1. *In variant* $\overline{\mathsf{WO}}$ *rendezvous is not feasible.*

Note that rendezvous can be obtained in more specific topologies. We discuss some cases below. It is worth to mention that the rendezvous algorithm for trees sketched in the introduction does not work when different maps are in use, as the centres computed for different maps may not coincide.

Lemma 4. *In variant* $\overline{\mathsf{WO}}$*, if network G is a path rendezvous can be solved optimally.*

Proof. Since $G_A \subseteq G_B \subseteq G$, also both G_A and G_B are paths. Each robot adopts the following strategy: *from the starting node s_X, go to an arbitrary endpoint of the path and then walk along all the edges to reach the other endpoint*. Due to the linear structure of the maps and inclusion assumption the robots must eventually meet. The complexity of rendezvous is trivially bounded by $O(N)$, and it is optimal in view of the lower bound from Lemma 1. □ □

Lemma 5. *In variant* $\overline{\mathsf{WO}}$*, if network G is a star graph rendezvous can be solved optimally.*

Proof. As $G_A \subseteq G_B \subseteq G$ then each of G_A and G_B can be a star, a single edge, or just a node. Each robot adopts the following strategy: *from the starting node s_X, visit each leaf, and finally stop at the centre*. In the degenerate case of a single edge, the robot can arbitrarily choose one node as the centre and apply the same strategy. If G_A is a node, the robot cannot move. Observe that either robots meet at a non-central node while attempting to enter the same edge, or they meet in the centre of the star, eventually. The complexity refers to a single traversal of the star and is bounded by $O(N)$. □ □

The next result affirms that in case the input map is a cycle the rendezvous problem cannot be solved in the $\overline{\mathsf{WO}}$ variant. In fact, cycles will play the central role in discussion on how the rendezvous complexity changes in the relevant variants.

Lemma 6. *In variant* $\overline{\mathsf{WO}}$*, if G is a cycle rendezvous cannot be resolved.*

Proof. Consider the case with $G_A = G_B$ both being a cycle with an even number of nodes. Assume an instance where the two robots lie at some antipodal nodes of the cycle. The adversary can force a symmetric behaviour of the two robots. That is, whatever one robot does according to the provided algorithm, the other makes exactly the same symmetric move. As robots are always located at some antipodal positions the meeting will never take place. □ □

Consider now the subset $I \subseteq \overline{\mathsf{WO}}$ containing all instances with $G_A = G_B$. If $I' \subset \mathsf{W}\overline{\mathsf{O}}$ contains all instances with $w_A = w_B$ drawn from I, we conclude using Theorem 1 that also in variant $\mathsf{W}\overline{\mathsf{O}}$ rendezvous is not always feasible.

Theorem 2. *In variant* $\mathsf{W}\overline{\mathsf{O}}$ *rendezvous with maps is not always feasible.*

The following lemma provides a feasibility results for variant $\mathsf{W}\overline{\mathsf{O}}$ when $w_A < w_B$ and the topology of G is restricted to cycles.

Lemma 7. *In variant* $\mathsf{W}\overline{\mathsf{O}}$*, if G is a cycle and $w_A < w_B$ then there exists an algorithm that allows robots to meet along walks of length $O(N \cdot |b_A|)$, where b_X is the binary representation of weight w_X.*

Proof. Since G is a cycle and $G_A \subseteq G_B \subseteq G$ then G_X is either a path or a cycle.

If G_X is a path then r_X applies the strategy provided in the proof of Lemma 4: *from s_X, r_X goes to an arbitrary endpoint of the path and then walk along all the edges to reach the other endpoint*. If G_X is a cycle, the algorithm works as follows. *Consider the binary representation b_X of w_X. Initially, robot r_X traverses the whole cycle (returning to s_X) in any direction; then, for each bit of b_X and starting from the least significant bit: if the current bit is 1, the robot performs a complete visit of the cycle in one direction, if the bit is 0, then the robot does the same in the opposite direction.*

If G_A is a path, the two robots meet within the first two visits of the cycle made by r_B, hence with a trajectory of length at most $2N$. If G_A is a cycle and $w_A < w_B$, the two trajectories differ as either (1) b_A and b_B have different sizes or (2) they differ for on least one bit. In the first case, r_B traverses G more times than r_A if they do not meet before, so they must meet eventually. In the second case, they robots traverse the cycle in the opposite directions at least once, and this is enough to force their meeting. The complexity of this algorithm is $O(N \cdot |b_A|)$, as $|b_A| \leq |b_B|$. □ □

4 A $O(N \log N)$ algorithm for variant WO

In [10], the authors define an algorithm for the case of synchronous robots that solves the rendezvous in variant WO utilising trajectories of length $O(N)$. A similar technique for asynchronous robots leads to trajectories of length $O(N^2)$. In what follows we propose a novel algorithm for asynchronous robots with the complexity $O(N \log N)$. The new algorithm is based on new techniques an it requires better understanding of the considered problem.

We start by observing that in variant WO one can define the total order $\prec^{\mathsf{WO}}$ on edges in E, where $G = (V, E)$ is the input network. This ordering is defined as follows: edges are first ordered according to their (increasing) weights, and in case of ties edges with smaller endpoints are earlier in the order. Formally, given two edges $e' = (v_i, v_j)$ and $e'' = (v_{i'}, v_{j'})$ then $e' \prec^{\mathsf{WO}} e''$ if and only if (1) $w(e') < w(e'')$, or (2) $w(e') = w(e'')$ and $\min(i, j) < \min(i', j')$.

Let $E = \{e_1, e_2, \ldots, e_m\}$ where $e_i \prec^{\mathsf{WO}} e_{i+1}$, for each $i = 1, 2, \ldots, m-1$ (i.e., indeces are consistent with the order $\prec^{\mathsf{WO}}$). Hence, if $G(i)$ is the subgraph of G induced by edges $e_i, e_{i+1}, \ldots, e_m$, the following properties hold: (1) $G(i)$ may be disconnected, and (2) $G(i+1)$ is a subgraph of $G(i)$.

Notice that the same notation adopted for elements of E is used to refer to edges in a map G_X, that is, if $E_X = \{e^X_1, e^X_2, \ldots, e^X_{m_X}\}$, then $e^X_i \prec^{\mathsf{WO}} e^X_{i+1}$ for each $i = 1, 2, \ldots, m_X - 1$.

We now introduce a rendezvous method called *two-steps approach. In the first step, a rendezvous algorithm $\mathcal{A}$ reduces the search space by computing a convenient sub-map $H_X \subseteq G_X$. In the second, $\mathcal{A}$ instruct each robots to meet inside H_X.*

The intuition behind this approach is *the smaller/simpler the search space, rendezvous becomes more efficient*. According to Lemma 1, H_X must contain all n_X nodes of G_X, thus the search space reduction can only affect edges from G_X. Also, since H_X must be connected, it contains at least $n_X - 1$ edges in the form of a spanning tree of G_X.

The search space reduction in variant WO is given below. Please note, this method cannot be used in the other three variants since it relies on order $\prec^{\mathsf{WO}}$ allowing to create the spanning tree T_X.

Definition 1. *Consider variant* WO *with maps G_X. Denote by T_X the* maximal spanning tree of G_X *obtained by Kruskal's algorithm, where edges are drawn in the reverse order to $\prec^{\mathsf{WO}}$.*

The following lemma determines a relationship between the maximal spanning trees T_A and T_B.

Lemma 8. *T_A is a subtree of T_B.*

Proof. In this variant $E_A \subseteq E_B$. Moreover, if there exists an edge $e \in T_B$ whose endpoints are both in V_A but $e \notin T_A$, then for any $e' \in T_A$ we have $e \prec^{\mathsf{WO}} e'$. Thus by applying the Kruskal's algorithm according to the reverse order to $\prec^{\mathsf{WO}}$, all edges selected in T_A will be also edges of T_B. □ □

In Fig. 1 we present a pseudo-code of procedure MAKEWALK adopting complete walk along edges of tree T_X. In particular, given T_X and a starting node s_X, by calling MAKEWALK(T_X, s_X) we obtain a walk W_X that starts at s_X, passes through all the edges of the tree (in each direction in the form of well defined Euler tour), and finishes at s_X. This property is crucial for any rendezvous algorithm based on traversing T_X several times. The following lemma provides a useful relationship between W_A and W_B. An example of application of procedure MAKEWALK is shown in Fig. 5.

Lemma 9. *Let $W_A =$ MAKEWALK(T_A, s_A), and $W_B =$ MAKEWALK(T_B, s_B). Then, $W_A \subseteq 2 \cdot W_B$, where $2 \cdot W_B$ is the concatenation of two occurrences of W_B.*

Procedure: MAKEWALK
Input : Tree T_X, initial robot's position s_X
Output : A walk W starting at s_X and passing through all nodes in T_X.

Let $N_{T_X}(s_X) = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$, such that $i_1 < i_2 < \ldots < i_k$;
$W =$ `list()` ; `// W is initialised as an empty list`
for $1 \leq j \leq k$ **do**
  Let $e = (s_X, v_{i_j})$;
  W.`append`(e).`concat`(MAKESUBWALK(T_X, v_{i_j}, s_X));
return W;

Procedure: MAKESUBWALK
Input : Tree T_X, current node s, previous node $f \in N_{T_X}(s)$
Output : A closed walk W starting and finishing in s, passing through all nodes in $T_X \setminus T_f$, where T_f is the maximal subtree of T_X rooted at f.

Let $N_{T_X}(s) = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$, such that $i_1 < i_2 < \ldots < i_k$;
Let $succ(v_{i_j}) = \begin{cases} v_{i_{j+1}} & \text{if } j < k \\ v_{i_1} & \text{otherwise} \end{cases}$
Let $next = succ(f)$;
Let $e' = (s, next)$ and $e'' = (s, f)$;
$W =$ `list`(e') ; `// W is initialised as a list containing just e'`
for $k - 1$ *times* **do**
  W.`concat`(MAKESUBWALK($T_X, next, s$)).`append`(e'');
  $next = succ(next)$;
return W;

Figure 1: Procedure MAKEWALK executed by robot $r_X \in \{r_A, r_B\}$ starting on node s_X of T_X. The requested walk W_X starting from and ending at the initial robot's position s_X, and containing all edges in T_X is obtained by exploiting the recursive Procedure MAKESUBWALK.

Proof. Assume first $s_A = s_B$. From Lemma 8 we have that $T_A \subseteq T_B$. Procedure MAKEWALK ensures that also the ordered tree T_A is a subtree of the ordered tree T_B, that is, nodes in the walks maintain their relative ordering. It follows that $W_A \subseteq W_B$. Since in general $s_A \neq s_B$ and the length of W_X is $2n_X - 3$, then W_B may contain a suffix $(v_j^A, v_{j+1}^A), (v_{j+1}^A, v_{j+2}^A), \ldots, (v_{2n_A-4}^A), v_{2n_A-3}^A)$ of W_A before its prefix $(v_1^A, v_2^A), (v_2^A, v_3^A), \ldots, (v_{j-2}^A, v_{j-1}^A)$, for some $1 < j < 2n_A - 2$. However, by traversing W_B twice, we can guarantee rendezvous by visiting W_A in the right order at least once. ☐ ☐

Algorithm WO-ASYNCH (cf Fig. 2) exploits Procedure MAKEWALK to build complete walks that fulfill condition of Lemma 9. The following theorem states that rendezvous in variant WO can be solved with complexity $O(N \log N)$, for the input network G with an arbitrary topology.

Theorem 3. *In variant* WO, *for any network G, Algorithm* WO-ASYNCH *guarantees rendezvous along a trajectory of length* $O(N \log N)$.

Proof. Algorithm WO-ASYNCH can be divided into four parts: (1) in the first two lines the spanning tree T_X is computed along with its integer logarithmic size (i.e., $k_X = \lceil \log |T_X| \rceil$), (2) Line 4, where the walk $W_X = \text{MAKEWALK}(T_X, s_X)$ is computed, (3) the block of Lines 5–14, where a target t_X is computed, and (4) block of Lines 15–18, where the complete walk W_X^+ is computed and performed. Such a walk W_X^+ consists of $2k_X$ concatenations of W_X plus a sub-sequence of W_X (i.e., the *final step*) needed to reach the target t_X. We now analyze two cases, according to the sizes k_A and k_B.

- Case $k_B > k_A$. We show that W_A^+ is a sub-walk of W_B^+, and hence from Lemma 3 the claim holds. In W_A^+ the sequence W_A is repeated $2k_A$ times plus a subsequence of W_A (due to the final step). In W_B^+, the sequence W_B is repeated $2k_B \geq 2(k_A + 1)$ times, which is at least $2k_A + 2$ times. From Lemma 9, the first two repetitions of

Algorithm: WO-ASYNCH

Input : Map $G_X = (V_X, E_X)$, starting node s_X, robot's weight w_X

`/* Part (1): compute` T_X `and its integer logarithmic size */`

1 compute the maximal spanning tree T_X (by using the ordering $\prec^{\mathsf{WO}}$);

2 **if** $|T_X| == 1$ **then exit** ;

3 let $k_X = \lceil \log |T_X| \rceil$;

`/* Part (2): compute walk` W_X `*/`

4 let $W_X = \text{MAKEWALK}(T_X, s_X)$;

`/* Part (3): compute target` t_X `*/`

5 let $i_X = \text{argmin}_i\{w(e_i^X) \mid e_i^X \in T_X\}$;

6 let $j = nil$;

7 **foreach** i *in* $(i_X + 1, i_X + 2, \ldots, m_X)$, *in order* **do**

8   let $T(i)$ be a largest subtree of T_X induced by nodes in $G(i)$;

9   **if** $k_X > \lceil \log |T(i)| \rceil$ **then**

10         $j = i - 1$;

11   **break**

12 **if** $j == nil$ **then** $j = m_X$;

13 let e be any edge in $T(j)$ having maximum order ;

14 let t_X be the endpoint of e with largest index ;

`/* Part (4): compute walk` W_X^+ `by using` W_X `and` t_X `*/`

15 **while** *not (* W_X *has been fully traversed* $2 \cdot k_X$ *times* $\vee$ *rendezvous is accomplished)* **do**

16   traverse the next edge in W_X

17 **while** *not (* r_X *is on* t_X $\vee$ *rendezvous is accomplished)* **do**

18   traverse the next edge in W_X

Figure 2: Algorithm WO-ASYNCH executed by robot $r_X \in \{r_A, r_B\}$.

W_B assure that W_A is contained in $2W_B$, that is a subsequence of W_B^+. It follows that $2k_A + 2$ sequences of W_B include $2k_A + 1$ sequences of W_A. Since W_A^+ is a subsequence of $2k_A + 1$ repetitions of W_A, then W_A^+ is contained in W_B^+.

- Case $k_B = k_A$. We show that robots r_A and r_B can select a common node t_X to rendezvous. When r_B executes Part (3) of the algorithm it computes a tree denoted as $T(j)$, which corresponds to the smallest subtree of T_B having size k_B. For r_B, tree $T(j)$ represents T_A according to the assumption $k_B = k_A$. For instance, in Fig. 6, T_1 represents T_B and T_3 represents $T(j) = T_A$. We now prove that there is exactly one occurrence of T_A in T_B, and the following relationships hold:

  - $|T_B| \geq |T_A|$;
  - $\lceil \log(|T_B|) \rceil = \lceil \log(|T_A|) \rceil$.

  Denoting by n the integer value $\lceil \log(|T_B|) \rceil = \lceil \log(|T_A|) \rceil$ we can represent $\log(|T_B|)$ and $\log(|T_B|)$ as follows:

  $$\log(|T_B|) = (n-1) + b, \ \log(|T_A|) = (n-1) + a, \ \text{with } 0 < a < b \leq 1.$$

  One can observe that $\log(|T_B|) - \log(|T_A|) = b - a < 1$, which in turns implies $\log(|T_B|/|T_A|) < 1$, $|T_B|/|T_A| < 2$, and finally the required relationship $|T_B| < 2|T_A|$. This relationship implies that if r_B selects at Lines 13 and 14 the largest in order edge e belonging to $T(j)$, and node t_X as the endpoint of e with the largest index, then the same target node will be selected by both r_A and r_B (i.e., $t_A = t_B$).

Summarising, if $k_B > k_A$, the algorithm forces robots to meet during the $2k_B$ repetitions of walk W_B. And if $k_B = k_A$, the algorithm forces robots to meet at $t_A = t_B$. Concerning the complexity, the trajectory of each robot is of length at most $2 \cdot k_B \cdot |T_B| = 2 \cdot \lceil \log |T_B| \rceil \cdot |T_B|$. Thus the total complexity of rendezvous is $O(N \log N)$. □ □

Algorithm: TREE-$\overline{\mathsf{WO}}$-ASYNCH
Input : Map defined by the tree $T_X = (V_X, E_X)$, starting node s_X

let $k_X = |T_X|$;
let $W_X = \text{MAKEWALK}(T_X, s_X)$;
let t_X be the node in V_X with maximum order ;
while *not (W_X has been fully traversed $2 \cdot k_X$ times $\vee$ rendezvous is accomplished)* **do**
  traverse the next edge in W_X
while *not (r_X is on t_X $\vee$ rendezvous is accomplished)* **do**
  traverse the next edge in W_X

Figure 3: Algorithm TREE-$\overline{\mathsf{WO}}$-ASYNCH executed by robot $r_X \in \{r_A, r_B\}$.

Theorem 3 indicates that Algorithm WO-ASYNCH is almost optimal as it solves rendezvous with trajectories of length $O(N \log N)$, and according to Lemma 1 the relevant lower bound is $\Omega(N)$. In terms of further improvements one could proceed along two different directions. One could try to find a more efficient algorithm for an arbitrary topology, or focus on some restricted classes of graphs. With respect to the latter, as the currently best rendezvous algorithm relies on spanning trees, the restricted cases would likely have to refer to sub-classes of trees. And indeed observe that the results provided in Section 3 for path graphs and star graphs also hold in variant WO.

5 Algorithms for variant $\overline{\mathsf{WO}}$

Concerning variant $\overline{\mathsf{WO}}$, in [10] one can find a rendezvous algorithm with double exponential (in N) complexity. Here we improve this result in specific classes of graphs.

We start by observing that in this variant it is possible to define a total ordering $\prec^{\overline{\mathsf{WO}}}$ on edges in E, where $G = (V, E)$ is the input network. This ordering is defined as follows: edges are ordered by utilising the total ordering of nodes. Formally, given two edges $e' = (v_i, v_j)$ and $e'' = (v_{i'}, v_{j'})$ such that $v_i < v_j$ and $v_{i'} < v_{j'}$ then $e' \prec^{\overline{\mathsf{WO}}} e''$ if and only if (1) $v_i < v_{i'}$, or (2) $v_i = v_{i'}$ and $v_j < v_{j'}$.

Observe that even if we have the total order $\prec^{\overline{\mathsf{WO}}}$, in variant $\overline{\mathsf{WO}}$ we cannot use the two-steps approach proposed in Section 4. In fact, if we compute again the maximal spanning tree (say T_X) of G_X by using the Kruskal's according to the reverse ordering of $\prec^{\overline{\mathsf{WO}}}$, the required property $T_A \subseteq T_B$ is not present any longer (it follows from different properties of maps in variants $\overline{\mathsf{WO}}$ and WO).

Nevertheless, one can adopt the two-steps approach in special classes of maps, including trees. Algorithm TREE-$\overline{\mathsf{WO}}$-ASYNCH shown in Fig. 3 can be used to solve rendezvous using trajectories of polynomial length.

Theorem 4. *In variant $\overline{\mathsf{WO}}$, when G is a tree Algorithm* TREE-$\overline{\mathsf{WO}}$-ASYNCH *allows robots to meet along a trajectory of length $O(N^2)$.*

Proof. Since $G_A \subseteq G_B \subseteq G$ then both G_A and G_B are trees. Hence, in the remainder, we denote the generic map as T_X.

Algorithm TREE-$\overline{\mathsf{WO}}$-ASYNCH computes the following: (1) the size $k_X = |T_X|$ of tree T_X, (2) the walk $W_X = \text{MAKEWALK}(T_X, s_X)$, (3) the target t_X as the node in V_X with maximum order, and (4) the complete walk W_X^+ consisting of $2k_X$ concatenations of W_X plus a sub-sequence of W_X needed to reach the target t_X.

We consider two cases reflecting on the relationship between k_A and k_B. If $|k_B| > |k_A|$, then walk W_B is repeated at least $2(k_A + 1) = 2k_A + 2$ times inside W_X^+. According to Lemma 9, we have $W_A \subseteq 2 \cdot W_B$ with W_A and W_B obtained by Procedure MAKEWALK. This means that we are in the same situation as in the respective case of the proof of Theorem 3, and hence the algorithm ensures that robots eventually meet. If $|k_B| = |k_A|$, the target t_X trivially fulfills $t_A = t_B$ and hence the rendezvous is eventually accomplished.

Concerning the complexity, the trajectory of each robot is of length at most $2 \cdot k_B \cdot |T_B| = 2 \cdot |T_B| \cdot |T_B|$ resulting in the total complexity $O(N^2)$. □ □

In the remaining, we propose efficient rendezvous algorithms for some other restricted topologies.

Lemma 10. *In variant* $\overline{\mathsf{W}}\mathsf{O}$*, when* G *is a cycle one can design an optimal rendezvous algorithm.*

Proof. Since G is a cycle and $G_A \subseteq G_B \subseteq G$ then each map G_X is either a path or the whole cycle. The rendezvous algorithm adopts the following strategy.

- If G_X is a cycle, robot r_X starts at s_X, and makes a complete walk in arbitrary direction visiting all nodes before returning to s_X. Then, r_X walks to the largest in provided order node t_X.
- If G_X is a path, r_X applies the strategy utilised in Lemma 4, i.e., robot r_X visits first an arbitrary endpoint of the path, then walks to the opposite endpoint on this path.

It is easy to see that robots do meet eventually, either on the final target node t_X or because r_B overpasses r_A. In both cases, the complexity is bounded by $O(N)$. □ □

Lemma 11. *In variant* $\overline{\mathsf{W}}\mathsf{O}$*, if both* G_A *and* G_B *are complete graphs (or complete bipartite graphs), there exists rendezvous algorithm with the complexity* $O(N^3)$*.*

Proof. Assume that both G_A and G_B are complete graphs. Each robot r_X computes its walk (rendezvous trajectory) W_X as follows:

1. Assume robot r_X is initially located at $s_X = v_i$, which becomes a *base node*.
2. From the current base node v_i, r_X visits back and forth all its neighbours starting from v_{n_X} down to v_{i+2}, and then it moves to the next base node v_{i+1} (in the periodic order);
3. Robot r_X repeats the same strategy until all nodes on its map served as base nodes.

On the basis of W_X, robot r_X computes the complete walk W_X^+ consisting of k_X concatenations of W_X plus a sub-sequence of W_X needed to reach the target node t_X which is the largest in the provided order. We now prove that if the two robots visit their own maps adopting W_X^+, the rendezvous is accomplished. We consider two cases based on the sizes of k_A and k_B. If $|k_B| > |k_A|$, by construction of W_X^+ we get $W_A \subseteq W_B$, and due to Lemma 3 rendezvous must be accomplished. If $|k_B| = |k_A|$, the thesis trivially follows since the target t_X are the same, $t_A = t_B$. Since the trajectory of each robot is at most $(k_X + 1) \cdot |W_X|$, and $|W_X| = O(N^2)$, the total complexity is $O(N^3)$.

One can observe that the above algorithm can be easily adapted when both G_A and G_B are complete bipartite graphs. □ □

6 Conclusion

We studied deterministic rendezvous of two asynchronous robots in the network modelled by graphs with restrictions imposed on edges. The restrictions prevent robots from visiting certain parts of the network. We considered four variants based on all possible combinations of presence/absence of two properties: (1) coherent ordering of nodes and (2) weighted robots/edges. We provided some impossibility results, lower bounds, and efficient algorithmic solutions. Two important problems remain open. The first is to establish whether our algorithm in variant WO is optimal. The second is to decide whether there exists a rendezvous algorithm in variant $\overline{\mathsf{W}}\mathsf{O}$ with the polynomial (in N) complexity, or the exponential approach provided in [10] cannot be improved.

References

[1] P. Borowiecki, S. Das, D. Dereniowski, L. Kuszner. Distributed Evacuation in Graphs with Multiple Exits. In *Proc. 23rd Int.'l Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 9988 of *LNCS*, pages 228–241. Springer, 2016.

[2] J. Chalopin, Y. Dieudonné, A. Labourel, and A. Pelc. Fault-tolerant rendezvous in networks. In *Proc. 41st Int.'l Colloquium on Automata, Languages, and Programming (ICALP)*, volume 8573 of *LNCS*, pages 411–422. Springer, 2014.

[3] J. Chalopin, Y. Dieudonné, A. Labourel, and A. Pelc. Rendezvous in networks in spite of delay faults. *Distributed Computing*, 29(3):187–205, 2016.

[4] J. Czyzowicz, A. Kosowski, and A. Pelc. How to meet when you forget: log-space rendezvous in arbitrary graphs. *Distributed Computing*, 25(2):165–178, 2012.

[5] J. Czyzowicz, A. Kosowski, and A. Pelc. Time versus space trade-offs for rendezvous in trees. *Distributed Computing*, 27(2):95–109, 2014.

[6] J. Czyzowicz, A. Labourel, and A. Pelc. How to meet asynchronously (almost) everywhere. *ACM Trans. Algorithms*, 8(4):37:1–37:14, 2012.

[7] G. De Marco, L. Gargano, E. Kranakis, D. Krizanc, A. Pelc, and U. Vaccaro. Asynchronous deterministic rendezvous in graphs. *Theor. Comput. Sci.*, 355(3):315–326, 2006.

[8] A. Dessmark, P. Fraigniaud, and A. Pelc. Deterministic rendezvous in graphs. In *Proc. 11th Annual European Symp. on Algorithms (ESA)*, volume 2832 of *LNCS*, pages 184–195. Springer, 2003.

[9] Y. DieudonnÃ©, A. Pelc, and V. Villain:. How to Meet Asynchronously at Polynomial Cost. *SIAM J. Comput.* 44(3):844–867, 2015.

[10] A. Farrugia, L. Gasieniec, L. Kuszner, and E. Pacheco. Deterministic rendezvous in restricted graphs. In *Proc. 41st Int.'l Conf. on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, volume 8939 of *LNCS*, pages 189–200. Springer, 2015.

[11] P. Flocchini, N. Santoro, G. Viglietta, and M. Yamashita. Rendezvous with constant memory. *Theor. Comput. Sci.*, 621:57–72, 2016.

[12] Z. Gu, Y. Wang, Q.S. Hua, and F.C.M Lau, Blind Rendezvous Problem, Rendezvous in Distributed Systems. Springer, Singapore, 2017.

[13] A. Pelc. Deterministic rendezvous in networks: A comprehensive survey. *Networks*, 59(3):331–347, 2012.

[14] G. Viglietta. Rendezvous of two robots with visible bits. In *Proc. 9th Int.'l Symp. on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics, (Algosensors)*, volume 8243 of *LNCS*, pages 291–306. Springer, 2013.

Appendix: Figures

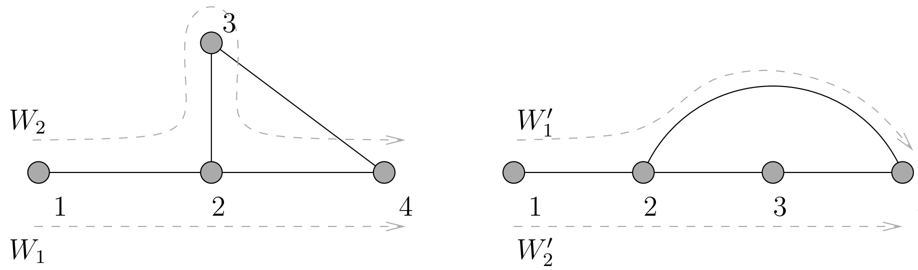

Figure 4: Examples of sub-walk: $W_1 = ((1,2),(2,4))$ is a sub-walk of $W_2 = ((1,2),(2,3),(3,2),(2,4))$. Contrary, $W'_1 = ((1,2),(2,4))$ is not a sub-walk of $W'_2 = ((1,2),(2,3),(3,4))$.

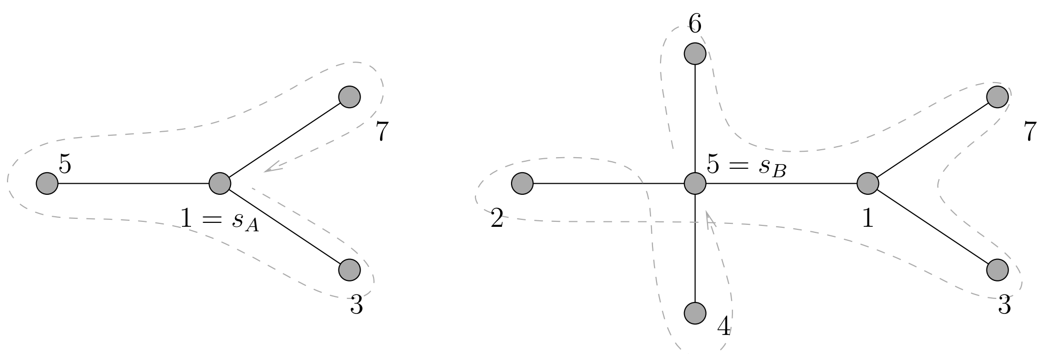

Figure 5: Examples of walks obtained by executing Procedure MAKEWALK:
$W_A = ((1,3),(3,1),(1,5),(5,1),(1,7),(7,1))$ and $W_B = ((5,6),(6,5),(5,1),(1,7),(7,1),(1,3),(3,1),(1,5),(5,2),(2,5),(5,4),(4,5))$.
Notice that
$2 \cdot W_B = ((5,6),(6,5),(5,1),(1,7),(7,1),$**(1,3)**,**(3,1)**,**(1,5)**,$(5,2),(2,5),(5,4),(4,5),(5,6),(6,5),$**(5,1)**,**(1,7)**,**(7,1)**,$(1,3),$
$(3,1),(1,5),(5,2),(2,5),(5,4),(4,5))$, and the sequence of edges forming W_A is highlighted in boldface inside $2 \cdot W_B$.

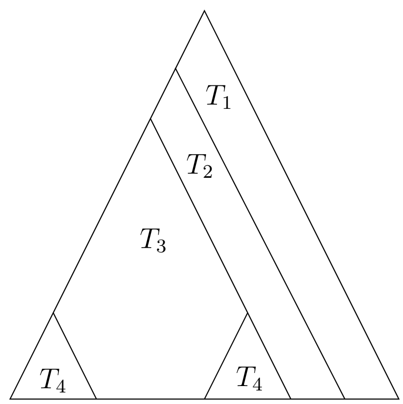

Figure 6: Visualisation of arguments used in Theorem 3. We assume that $w_1 < w_2 < w_3 < w_4$ are four possible edge weights, and in the picture T_i, $1 \leq i \leq 4$, represents a tree containing edges with weights $w_i, w_{i+1}, \ldots, w_4$ only. Hence, $T_4 \subset T_3 \subset T_2 \subset T_1$.