Appendix A. Formal Proofs

Throughout this paper, we provided proof sketches to convey the gist of the proof when presenting the full proof would break flow of the prose. In this Appendix, we provide the formal proofs for all such Theorems.

A.1 Preliminaries

In this section, we provide present definitions and lemmas that will be useful in our proofs of correctness for the \( \Delta \)DPC, and \( \Delta \)PPC algorithms. We begin by defining a key relation about elimination orderings. Once we have defined this relationship, we will prove some properties about this relationship that will be useful for proving that our algorithms correctly establish PPC.

**Definition 1.** Given a graph \( \mathcal{G} = (V, E) \) and a total ordering \( o \) over \( V \)—that is \( \forall v_x, v_y \in V \) such that \( x \neq y \) implies \( v_x \prec o v_y \lor v_y \prec o v_x \)—let \( \mathcal{G}_o^\Delta = (V, E \cup E_o^\Delta) \) be the graph that results from triangulating graph \( \mathcal{G} \) by eliminating vertices in order \( o \) and adding fill edges \( E_o^\Delta \).

**Definition 2.** Given a graph \( \mathcal{G} = (V, E) \) and (total) elimination order \( o \), we define the precedence relation \( \prec o^\Delta \), where \( v_x \prec o^\Delta v_y \) if and only if, at the time of \( v_x \)'s elimination, \( v_y \) shares an edge with \( v_x \) and has not been eliminated. That is, \( v_x \prec o^\Delta v_y \iff (v_x \prec o v_y) \land (e_{xy}, e_{yz} \in E \cup E_o^\Delta) \).

**Definition 3.** We label lines 3-11 of the \( \Delta \)DPC algorithm as the ELIMINATE procedure. This procedure ELIMINATES a timepoint \( v_k \) after first using edges \( e_{ik} \) and \( e_{kj} \) to tighten (and when necessary, to add) each edge \( e_{ij} \) for every pair of non-eliminated, neighboring timepoints, \( v_i \) and \( v_j \).

**Definition 4.** We label lines 4-8 of the \( \Delta \)PPC algorithm as the REINSTATE procedure. This procedure REINSTATES a timepoint \( v_k \) by, for every pair of previously-reinstated, neighboring timepoints \( v_i \) and \( v_j \), tightening edge \( e_{ki} \) and edge \( e_{jk} \) with respect to \( e_{ij} \).

**Lemma 1.** Let \( o \) and \( o' \) be two distinct total orderings of the vertices, \( V \) for some graph \( \mathcal{G} = (V, E) \). If \( o' \) is consistent with the precedence relation \( \prec o^\Delta \), then \( \mathcal{G}_o^\Delta = \mathcal{G}_{o'}^\Delta \).

**Proof.** Assume \( \mathcal{G}_o^\Delta \neq \mathcal{G}_{o'}^\Delta \).

Since \( \mathcal{G}_o^\Delta = (V, E \cup E_o^\Delta) \) and \( \mathcal{G}_{o'}^\Delta = (V, E \cup E_{o'}^\Delta) \), if \( \mathcal{G}_o^\Delta \neq \mathcal{G}_{o'}^\Delta \) then \( E_o^\Delta \neq E_{o'}^\Delta \).

\( E_o^\Delta \neq E_{o'}^\Delta \) implies that there exists at least one edge \( e_{xy} \) such that, either \( e_{xy} \in E_o^\Delta \) and \( e_{xy} \notin E_{o'}^\Delta \), or \( e_{xy} \notin E_o^\Delta \) and \( e_{xy} \in E_{o'}^\Delta \).

WLOG, let \( e_{xy} \in E_o^\Delta \) be the first edge that is added to \( E_o^\Delta \) under elimination order \( o \) that is not added to \( E_{o'}^\Delta \) under \( o' \). In order for edge \( e_{xy} \) to be added under elimination order \( o \), there must be some vertex \( v_z \) such that it is eliminated prior to \( v_x \) and \( v_y \) and shares an edge with both \( v_x \) and \( v_y \) (\( e_{xz} \) and \( e_{yz} \) respectively) at the time of its elimination. By
definition, this implies \( v_z \prec_o v_x \) and \( v_z \prec_o v_y \). So, under the assumption that \( o' \) respects the precedence relation \( \prec_o \), \( o' \) eliminates \( v_x \) prior to \( v_y \). Since \( v_z \) is eliminated prior to \( v_x \) and \( v_y \) but no fill edge \( e_{xy} \) is added, at least one of \( e_{xz} \) or \( e_{xy} \) is absent at the time of \( v_z \)’s elimination under \( o' \). WLOG, assume \( e_{xz} \) is missing. If \( e_{xz} \) is missing at the time of \( v_z \)’s elimination it cannot be part of the original specification of \( G \), which implies it is a member of \( E_0^\triangle \). However, once \( v_z \) is eliminated, no new edge \( e_{xz} \) can ever be constructed, since fill edges are only ever added between non-eliminated vertices. Thus, either \( e_{xy} \) is not the first edge that is added to \( E_0^\triangle \) under elimination order \( o \) that is not added to \( E_0^\triangle \) under \( o' \), or \( o' \) does not respect the precedence relation \( \prec_o \), but both cases violate the assumptions. Therefore, since every time elimination order \( o \) adds a fill edge \( e \), it induces a new \( \prec_o \) relation, any other elimination order \( o' \) that also satisfies the relation \( \prec_o \) will also add the fill edge, implying \( E_0^\triangle \subseteq E_o^\triangle \).

Next we prove that \( E_o^\triangle \subseteq E_o' \), which mirrors the proof that \( E_o^\triangle \subseteq E_o' \). WLOG, let \( e_{xy} \in E_o^\triangle \) be the first edge that is added to \( E_o^\triangle \) under elimination order \( o' \) that is not added to \( E_0^\triangle \) under \( o \). In order for edge \( e_{xy} \) to be added under elimination order \( o' \), there must be some vertex \( v_z \) such that it is eliminated prior to \( v_x \) and \( v_y \) and shares an edge with both \( v_x \) and \( v_y \) (\( e_{xz} \) and \( e_{xy} \) respectively) at the time of \( v_z \)’s elimination. Since \( e_{xy} \) is the first edge added to \( E_o^\triangle \) under elimination order \( o' \) that is not added to \( E_0^\triangle \) under \( o \), and also since no edges can be added after the elimination of one of its endpoints, \( e_{xz} \) must already exist at the time of both \( v_x \)’s and \( v_z \)’s elimination under \( o \). However, since elimination order \( o \) does not add \( e_{xy} \), at least one of \( v_x \) or \( v_y \) is eliminated before \( v_z \) under \( o \). WLOG assume \( v_x \) is eliminated prior to \( v_z \). However, since at the time of \( v_x \) elimination, \( v_x \) and \( v_z \) share edge \( e_{xz} \) and so by definition \( v_x \prec_o v_z \). This contradicts the assumption that \( o' \) respects \( \prec_o \). Therefore, since the order that vertices that share edges are specified as part of \( \preceq_o \) by definition, if elimination order \( o' \) respects \( \prec_o \), \( E_o^\triangledown \subseteq E_o' \).

Since \( E_0^\triangle \subseteq E_o' \) and \( E_0^\triangledown \subseteq E_o' \), \( E_0^\triangle = E_0' \) which violates the assumption that \( G_0^\triangledown \neq G_o^\triangle \) since they only can differ in fill edges. Therefore, if \( o' \) is consistent with the precedence relation \( \prec_o \), then \( G_o^\triangledown = G_o^\triangle \).

**Lemma 2.** Let \( o \) be a total elimination order used to triangulate STN \( G \), resulting in graph \( G_o^\triangle \) and precedence relation \( \prec_o \). Any application of \( \triangle \text{DPC} \) that eliminates nodes with respect to the precedence relation \( \prec_o \) will have the same output as \( \text{DPC}(o, G_o^\triangledown) \).

**Proof.** By contradiction: Let \( G^\triangle \text{DPC} \) be the output of \( \triangle \text{DPC} \) and \( G^\text{DPC} \) be the output of DPC. Assume \( G^\triangle \text{DPC} \neq G^\text{DPC} \). By Lemma 1, \( G^\triangle \text{DPC} \) and \( G^\text{DPC} \) will contain the same edges. This implies for at least one edge \( e_{xy} \), \( w_{xy}^\triangle \text{DPC} \neq w_{xy}^\text{DPC} \).

**Part 1:** Suppose after applying both DPC and \( \triangle \text{DPC} \), there was an edge \( e_{xy} \), where \( w_{xy}^\text{DPC} < w_{xy}^\triangle \text{DPC} \) and, WLOG, this was the first edge that DPC tightened further than \( \triangle \text{DPC} \). This implies that for at least one vertex \( v_z \), DPC performs the update \( w_{xy}^\triangle \text{DPC} \leftarrow \min(w_{xy}^\text{DPC}, w_{xz}^\text{DPC} + w_{zy}^\text{DPC}) \) and either \( \triangle \text{DPC} \) does not, or if it does, \( \min(w_{xy}^\text{DPC}, w_{xz}^\text{DPC} + w_{zy}^\text{DPC}) < \min(w_{xy}^\triangle \text{DPC}, w_{xz}^\triangle \text{DPC} + w_{zy}^\triangle \text{DPC}) \). However, since DPC only performs the update \( w_{xy}^\text{DPC} \leftarrow \min(w_{xy}^\text{DPC}, w_{xz}^\text{DPC} + w_{zy}^\text{DPC}) \) if and only if edges exists between \( v_x, v_y \), and \( v_x \) and \( v_z \) is eliminated before \( v_x \) and \( v_y \), by definition, \( v_z \prec_o v_x \) and \( v_z \prec_o v_y \).
Since $\nabla DPC$ eliminates nodes with respect to the precedence relation $\prec_o$, $\nabla DPC$ must eliminate $v_z$ before eliminating $v_x$ and $v_y$, resulting in the update:

$$w_{xy}^{\nabla DPC} \leftarrow \min(w_{xy}^{\nabla DPC}, w_{xz}^{\nabla DPC} + w_{zy}^{\nabla DPC})$$

so unless the assumption that $\nabla DPC$ respects $\prec_o$ is violated, $\nabla DPC$ correctly applies the update.

Since $\nabla DPC$ correctly applies the update $w_{xy}^{\nabla DPC} \leftarrow \min(w_{xy}^{\nabla DPC}, w_{xz}^{\nabla DPC} + w_{zy}^{\nabla DPC})$, the only way that $w_{xy}^{DPC} < w_{xy}^{\nabla DPC}$ holds true after the update is if either $w_{xy}^{\nabla DPC} < w_{xy}^{DPC}$, or $w_{yz}^{\nabla DPC} < w_{yz}^{DPC}$ at the time the update is performed. But this violates the assumption that $w_{xy}^{DPC} < w_{xy}^{\nabla DPC}$ is the first update performed by $DPC$ that is never correctly performed by $\nabla DPC$.

Thus $DPC$ will never perform an update to the bound $w_{xy}^{DPC}$ of any edge $e_{xy}$ that will not also be applied by $\nabla DPC$, thus $w_{xy}^{DPC} \geq w_{xy}^{\nabla DPC}$.

**Part 2:** Suppose after applying both $DPC$ and $\nabla DPC$, there exists an edge $e_{xy}$, where $w_{xy}^{\nabla DPC} < w_{xy}^{DPC}$, that was the first edge that $\Delta DPC$ tightens further than $DPC$. This implies there must be some vertex $v_z$ such that $\nabla DPC$ eliminates it prior to $v_x$ and $v_y$ and that shares edges with both $v_x$ and $v_y$ with tightened values $w_{xz}^{\nabla DPC}$ and $w_{zy}^{\nabla DPC}$ respectively. Further, at the time of $v_z$’s elimination, $\nabla DPC$ tightens the bound $w_{xy}^{\nabla DPC}$ using the rule $w_{xy}^{\nabla DPC} \leftarrow \min(w_{xy}^{\nabla DPC}, B_{xz}^{\nabla DPC} + w_{zy}^{\nabla DPC})$.

Since $w_{xy}^{\nabla DPC}$ is the first bound that $\nabla DPC$ tightens further than $DPC$ using elimination order $o$ and also since $DPC$ does not tighten the bounds of any edge after it eliminates one of its endpoints, $DPC$ will have already tightened $w_{xz}^{DPC}$ by the time it eliminates either $v_x$ or $v_z$ and $DPC$ will have already tightened $w_{xy}^{DPC}$ by the time it eliminates either $v_y$ or $v_z$. Thus, if $v_z$ appears before $v_x$ and $v_y$, $DPC$ would apply the update $w_{xy}^{DPC} \leftarrow \min(w_{xy}^{DPC}, B_{xz}^{DPC} + w_{zy}^{DPC})$ with $w_{xz}^{DPC} = w_{x}^{\Delta DPC}$ and $w_{zy}^{DPC} = w_{y}^{\Delta DPC}$, which is exactly the same update as $\nabla DPC$. Thus, $w_{xy}^{\nabla DPC} < w_{xy}^{DPC}$, $DPC$ must never apply the update, implying $v_z$ must appear after either $v_x$ or $v_y$ in $o$.

WLOG, assume $v_x$ appears before $v_z$ in $o$. However, as shown in Lemma 1, at the time of the elimination of $v_x$ or $v_z$, edge $e_{xz}$ must already exist, since edges are never added between eliminated vertices. Since $v_x$ and $v_z$ share an edge and $v_x$ appears before $v_z$ in $o$, by definition $v_x \prec_o v_z$. This contradicts the assumption that the order $\nabla DPC$ eliminates vertices respects $\prec_o$. Therefore, $w_{xy}^{\nabla DPC} \geq w_{xy}^{DPC}$.

**Conclusion:** Since both $w_{xy}^{\nabla DPC} \geq w_{xy}^{\nabla DPC}$ and $w_{xy}^{\nabla DPC} \geq w_{xy}^{DPC}$, then $w_{xy}^{\nabla DPC} = w_{xy}^{DPC}$. However this contradicts the assumption that $\nabla PPc \neq \nabla DPC$. Therefore, the output, $\nabla PPc$, of an application of $\nabla DPC$ will be the same as the output, $\nabla DPC$, of $PPc(\Delta G^o)$ if $\nabla DPC$ eliminates nodes with respect to the precedence relation $\prec_o$. \qed

**Lemma 3.** Let $o$ be a total elimination order used to triangulate STP $G$, resulting in graph $G^o$ and precedence relation $\prec_o$. Also let $G' = (V, E')$ be the output of $PPc(o, G^o)$. Then the output, $\nabla PPc$, of any application of the second phase of the $\nabla PPC$ algorithm that reinstates vertices in reverse $\prec_o$ order will be the same as the output, $\nabla PPC$, of applying $PPc(o, G^o)$.

**Proof.** When a vertex $v_x$ is reinstated, both $\nabla PPC$ and $PPc$ apply the following updates:
\[
\begin{align*}
\bullet & \quad w_{xi} \leftarrow \min(w_{xi}, w_{xj} + w_{ji}) \\
\bullet & \quad w_{xj} \leftarrow \min(w_{xj}, w_{xi} + w_{ij}) \\
\bullet & \quad w_{ix} \leftarrow \min(w_{ix}, w_{ij} + w_{jx}) \\
\bullet & \quad w_{jx} \leftarrow \min(w_{jx}, w_{ji} + w_{ix})
\end{align*}
\]

\(\forall i, j\) such that \(e_{xi} \in E'\), where \(v_x\) appears before \(v_i\) and \(v_j\) in \(o\). By Lemma 2, the call to \(\triangle\)PDC in line 2 will produce the same output as the call to DPC by the \(P^3C\) algorithm.

By contradiction: Assume that applying \(P^3C\) \((o, \Delta(o, G'))\) achieves a different output than an application of \(\triangle\)PDC to \(G'\) that reinstates vertices in reverse \(\prec^\Delta\) order does. Then there exists at least one pair of vertices, \(v_x\) and \(v_i\), where, WLOG, \(v_x\) appears before \(v_i\) in \(o\), such that \(w_{xi}^{P^3C} \neq w_{xi}^{\triangle\text{PDC}}\). So either \(w_{xi}^{P^3C} < w_{xi}^{\triangle\text{PDC}}\) or \(w_{xi}^{P^3C} > w_{xi}^{\triangle\text{PDC}}\). WLOG, let \(w_{xi}^{P^3C} \neq w_{xi}^{\triangle\text{PDC}}\) be the first such difference between \(G^{P^3C}\) and \(G^{\triangle\text{PDC}}\).

**Part 1:** Assume that after both \(P^3C\) and \(\triangle\)PDC are applied, \(w_{xi}^{P^3C} < w_{xi}^{\triangle\text{PDC}}\).

Thus \(P^3C\) applies some update \(w_{xi}^{P^3C} \leftarrow \min(w_{xi}^{P^3C}, w_{xj}^{P^3C} + w_{ji}^{P^3C})\) that \(\triangle\)PDC either does not apply or applies when \(w_{xi}^{P^3C} < w_{xi}^{\triangle\text{PDC}}\) or \(w_{ij}^{P^3C} < w_{ij}^{\triangle\text{PDC}}\).

Notice that the only time a bound \(w_{ij}^{P^3C}\) is updated during \(P^3C\) is when either \(v_i\) or \(v_j\) is being considered. Thus, any updates to \(w_{xi}^{P^3C}\), \(w_{xj}^{P^3C}\), or \(w_{ij}^{P^3C}\) must have occurred when processing either \(v_i\) or \(v_j\), both of which appear later than \(v_x\) in \(o\). If \(e_{xj} \in G'\) and \(v_x\) appears before \(v_i\) and \(v_j\) in \(\prec^\Delta\), thus \(\triangle\)PDC will also apply this update. Since we assumed this was the first time \(P^3C\) and \(\triangle\)PDC differed, neither \(w_{ij}^{P^3C} < w_{ij}^{\triangle\text{PDC}}\) nor \(w_{ij}^{P^3C} < w_{ij}^{\triangle\text{PDC}}\) can be true.

Thus, there is a contradiction, and so \(w_{xi}^{P^3C} \geq w_{xi}^{\triangle\text{PDC}}\).

**Part 2:** Assume that after both \(P^3C\) and \(\triangle\)PDC are applied, \(w_{xi}^{\triangle\text{PDC}} < w_{xi}^{P^3C}\).

Since \(w_{xi}\) is the first place that \(P^3C\) applies a different update than \(\triangle\)PDC, the difference cannot occur as a result of a tighter bound \(w_{ij}^{\triangle\text{PDC}} < w_{ij}^{P^3C}\) or \(w_{ij}^{\triangle\text{PDC}} < w_{ij}^{P^3C}\) at the time of the update \(w_{xj} \leftarrow \min(w_{xj}, w_{xj} + w_{ji})\). Thus, \(\triangle\)PDC must apply an update that \(P^3C\) does not apply, which can only occur in two cases.

**Case 1:** There exists some \(v_k\) that appears later than \(v_x\) in \(o\) such that \(v_x\) and \(v_k\) share an edge during \(\triangle\)PDC’s execution but not \(P^3C\). However, this violates Lemma 2.

**Case 2:** \(\triangle\)PDC reinstates some \(v_k\) that shares an edge with \(v_x\) before reinstating \(v_x\), but that appears earlier than \(v_x\) in \(o\). However, if \(v_k\) shares an edge with \(v_x\) and appears before \(v_x\) in \(o\), then \(v_x \prec^\Delta v_k\), which violates the assumption that \(\triangle\)PDC reinstates vertices in reverse \(\prec^\Delta\).

Therefore \(w_{xi}^{P^3C} \leq w_{xi}^{\triangle\text{PDC}}\).

**Conclusion:** Since, for the outputs of \(P^3C\) \((o, G')\) and \(\triangle\)PDC, \(w_{xi}^{P^3C} \geq w_{xi}^{\triangle\text{PDC}}\) and \(w_{xi}^{P^3C} \leq w_{xi}^{\triangle\text{PDC}}\), \(w_{xi}^{P^3C}\) must equal \(w_{xi}^{\triangle\text{PDC}}\) for all \(x, i\). Therefore the outputs of \(P^3C\) \((o, G')\) and \(\triangle\)PDC are identical. \(\square\)

So far, we have defined a key precedence relation of graph triangulations, \(\prec^\Delta\). We have shown that any elimination order \(d'\) that respects this precedence relation will result in the same triangulated graph. Further, we have shown that any application of the \(\triangle\text{PDC}\)
(\(\Delta\text{DPC}\)) algorithm that respects the precedence relation \(\prec_{\Delta}\) as it eliminates and reinstates vertices and tightens bounds will calculate exactly the same PPC (DPC) STN as applying the P\(^3\text{C}\) (DPC) algorithm using \(o\). Notice that since we proved this for each phase of the P\(^3\text{C}\) algorithm independently, as long both phases of \(\Delta\text{PPC}\) respect \(\prec_{\Delta}\), the total order in which it reinstates vertices in the two phases can be different.

We must now prove that both our \(\Delta\text{PPC}\) and our \(\Delta\text{DPPC}\) algorithms correctly apply \(\Delta\text{DPC}\) and \(\Delta\text{PPC}\) respectively to calculate a PPC STP instance.

**A.2 The \(\Delta\text{DPC}\) Algorithm Proof of Correctness**

This theorem proves the correctness of the \(\Delta\text{DPC}\) algorithm (Algorithm 7 on page 119). Note, this proof builds on definitions and properties established in Section A.1.

**Theorem 1.** \(\Delta\text{DPC}\) correctly establishes DPC on the multiagent STP.

**Proof.** Notice that the semantics of \(\Delta\text{DPC}\) dictate that each agent \(i\) eliminates its private timepoints \(V^i_P\) in some order \(o^i_P\) before eliminating its shared timepoints \(V^i_S\), which are eliminated in a globally consistent order \(o_S\). Despite the fact that agents eliminate timepoints concurrently, using a fine enough granularity of time, this implies that globally, all private timepoints are eliminated in some order \(o_D\), which respects the partial order \(o^i_P\wedge o^i_P\wedge\cdots\wedge o^i_P\wedge o_S\), where \(\wedge\) appends two orderings together. This proof proceeds to show that \(\Delta\text{DPC}\) establishes DPC on \(G\) by showing that it calculates the same result as DPC(\(o_D, G\)) by demonstrating that \(\Delta\text{DPC}\) correctly applies DPC with respect to \(\prec_{o_D}\).

We begin this proof by appealing to Lemma 2 which states that any application of \(\Delta\text{DPC}\) that respects precedence relation \(\prec_{o_D}\) achieves the same output as DPC(\(o_D, G\)). We show that, despite its concurrent execution, \(\Delta\text{DPC}\) eliminates vertices (and so applies \(\Delta\text{DPC}\)) in a way that respects precedence relation \(\prec_{o_D}\) and therefore achieves the same output as the same out as DPC(\(o_D, G\)). We do this by considering the elimination of some timepoint \(v^i_x\), where \(v^i_x\) belongs to agent \(i\).

Assume that there exists some \(v_y\) such that \(v_y \prec_{o_D} v^i_x\) but has not been eliminated by the time agent \(i\) eliminates \(v^i_x\).

**Case 1:** \(v^i_x\) and \(v_y\) belong to the same agent \(i\).

Notice \(v_y \prec_{o_D} v^i_x\) implies \(v_y \prec_{o_D} v^i_x\). However, if both \(v_y\) and \(v^i_x\) belong to agent \(i\), they must both appear in \(o^i_P\wedge o_S\) (constructed in lines 1 and 7) and therefore, by construction of \(o_D\), \(v_y \prec_{o_D} v^i_x\). This presents a contradiction since \(v_y \prec_{o_D} v^i_x\) is true if and only if agent \(i\) executing \(\Delta\text{DPC}\) eliminates \(v_y\) before eliminating \(v^i_x\), but we assumed agent \(i\) will have not eliminated \(v_y\) by the time it eliminates \(v^i_x\). Therefore, \(v_y\) must belong to some agent \(j\) where \(i \neq j\).

**Case 2:** \(v^i_x\) is private and \(v_y\) belongs to some agent \(j\) where \(i \neq j\).

Since \(v^i_x\) is private, by definition there can be no edge \(e_{xy} \in E\). Further, since \(v^i_x\) is private, all of its neighbors are local to agent \(i\), and since, by definition of \(o_D\), the only vertices that agent \(i\) eliminated at this point are also private, no fill edge between \(v^i_x\) and \(v_y\) could have been added. Therefore \(v_y\) must belong to agent \(i\). However, we have already shown that this can never be the case, thus establishing a contradiction. Therefore, if \(v^i_x\) is private, at the time that agent \(i\) executing \(\Delta\text{PPC}\) eliminates \(v^i_x\) there can exist no \(v_y\).
such that \( v_y \prec_{D} v^i_x \) but has not been eliminated. So for the assumption to hold, \( v^i_x \) must be shared, which brings us to the third and final case.

**Case 3:** \( v^i_x \) is shared and \( v_y \) belongs to some agent \( j \) where \( i \neq j \).

At this point, \( v^i_x \) and \( v_y \) must be shared, \( v_y \) must belong to some agent \( j \), where \( j \neq i \), and we assume both that \( v_y \prec_{D} v^i_x \) and agent \( i \) eliminates \( v^i_x \) before agent \( j \) eliminates \( v_y \). Because of the assumption that agent \( i \) eliminates \( v^i_x \) before agent \( j \) eliminates \( v_y \), the following sequence of events can never occur – agent \( j \) eliminates \( v_y \) – agent \( i \) synchronizes its view of the MaSTP – agent \( i \) eliminates \( v^i_x \).

However before the elimination of \( v^i_x \), agent \( i \) first has to obtain a lock on the shared elimination order (line 5). Thus, if \( v_y \) appears before \( v^i_x \), the agent \( i \) would learn of this in line 10. Line 11 would then ensure that agent \( i \) waits to receive all pertinent edge updates (w.r.t. \( v_y \)). Thus, agent \( i \) could never eliminate \( v^i_x \) at the same time as, or prior to \( v_y \), if \( v_y \) appears before it in \( \prec_{D} \).

Therefore, whether \( v^i_x \) is private or shared and \( v_y \) belongs to agent \( i \) or some other agent \( j \neq i \), D\( \triangle \)DPC correctly eliminates timepoints with respect to \( \prec_{D} \), and so by Lemma 2, calculates the same output as DPC\((O_D, G)\).

\[ \square \]

### A.3 D\( \triangle \)PPC is Deadlock Free

Here we prove Theorem 5, originally stated on page 123, which establishes that the D\( \triangle \)DPC algorithm (Algorithm 5 on page 123) is deadlock free.

**Theorem 5.** D\( \triangle \)PPC is deadlock free.

**Proof.** This is a continuation of the proof of Theorem 5 (page 123), which already established that line 1 is deadlock free. Notice that each agent reinstates nodes in reverse \( o_S \) order (line 3). By contradiction, assume line 8, which represents the only blocking communication in this algorithm, introduces a deadlock. This implies that there are two (or more) agents, \( i \) and \( j \), where \( i \neq j \) such that both agent \( i \) and agent \( j \) are simultaneously waiting for communication from each other in line 8. Thus, there exists a timepoint \( v^i_x \in V^i_X \cap V^j_L \) for which agent \( i \) is waiting to receive updated edges from agent \( j \), while there is also a \( v^j_y \in V^j_X \cap V^i_L \) for which agent \( j \) is waiting to receive updated edges from agent \( i \). Notice that \( v^i_k \) (the timepoint that agent \( i \) is currently considering) must appear before \( v^j_y \) (hence the need for blocking communication), but after \( v^i_x \) in agent \( i \)'s copy of \( o_S \), because otherwise agent \( i \) would have already sent agent \( j \) all edge updates pertaining to \( v^i_x \) (line 14) in the previous loop iteration in which \( v^i_x \) was reinstated. However, for the same reason, \( v^j_k \) (the timepoint that agent \( j \) is currently considering) must appear before \( v^i_x \) but after \( v^j_y \) in \( o_S \).

But this is a contradiction, because \( o_S \) is constructed in a way that consistently and totally orders all shared timepoints. This argument extends inductively to three or more agents, and so line 8 can also not be the cause of a deadlock. This is a contradiction.

Therefore the D\( \triangle \)PPC algorithm is deadlock free. \[ \square \]
A.4 The D\(\Delta\)PPC Algorithm Proof of Correctness

Here we prove Theorem 6, originally stated on page 124, which establishes the correctness of the D\(\Delta\)DPC algorithm (Algorithm 7 on page 119). Note, this proof builds on definitions and properties established in Section A.1 and Theorems 5 and 1.

**Theorem 6.** D\(\Delta\)PPC correctly establishes PPC on the MaSTN.

**Proof.** Notice that the semantics of D\(\Delta\)DPC dictate that each agent \(i\) eliminates its private timepoints \(V^i_P\) in some order \(o^i_P\), before eliminating its shared timepoints \(V^i_S\), which are eliminated in a globally consistent order \(o_S\). Despite the fact that agents eliminate timepoints concurrently, using a fine enough granularity of time, this implies that globally, all private timepoints are eliminated in some order \(o_P\), which respects the partial order \(o^i_P\forall i\) and \(o_S\). WLOG, let \(o_D = o^i_P\land o^i_P\land\cdots\land o^i_P\land o_S\), where \(\land\) appends two orderings together. This proof proceeds to show that D\(\Delta\)PPC establishes PPC \(G\) by showing that it calculates the same result as P\(3\)C \((o_D,G)\) by demonstrating that D\(\Delta\)PPC reinstates vertices (and so correctly applies DPPC) with respect to \(\prec_{o_D}\).

We start by acknowledging the proof of Theorem 1, which demonstrates that line 1 correctly establishes DPC.

By Lemma 3, if the reverse sweep of the \(\Delta\)PPC algorithm reinstates \(v^i_x\) after it reinstates \(v^i_y\) if \(v^i_x \prec_{o_D} v^i_y\), it achieves the same out as P\(3\)C \((o,G)\). We now show that, despite its concurrent execution, the last time D\(\Delta\)PPC reinstates \(v^i_x\) is after it reinstates \(v^i_y\) if \(v^i_x \prec_{o_D} v^i_y\).

By contradiction, assume agent \(i\) reinstates \(v^i_x\) (i.e., applies line 3-14 of the D\(\Delta\)PPC Algorithm) before \(v^i_y\), despite the fact that \(v^i_x \prec_{o_D} v^i_y\).

**Case 1:** \(v^i_x\) and \(v^i_y\) belong to the same agent \(i\).

Notice \(v^i_x \prec_{o_D} v^i_y\) implies \(v^i_x \prec_{o_D} v^i_y\). However, if both \(v^i_y\) and \(v^i_x\) belong to agent \(i\), they must both appear in \(o^i\) (i.e., \(o^i_P\land o_S\)) and therefore, by construction of \(o_D\), \(v^i_x \prec_{o^i} v^i_y\). However, line 3 explicitly reinstates nodes in reverse \(o^i\) order. Therefore, \(v^i_y\) must belong to some agent \(j\) where \(i \neq j\).

**Case 2:** \(v^i_y \in V^i_L\) for some agent \(j \neq i\).

\(v^i_x \prec_{o_D} v^i_y\) implies there is an edge between \(v^i_x\) and \(v^i_y\) at the time \(v^i_x\) is eliminated, which, by definition, implies \(v^i_y\) is not private. Further, if \(v^i_y\) is private, Theorem 1 states agent \(j\) can reason over it independently of agent \(i\). Thus, either way we have a contradiction, thus \(v^i_y\) cannot be private to some other agent \(j\).

**Case 3:** \(v^i_y \in V^i_L\), that is \(v^i_y \in V^i_L\) for some agent \(j \neq i\).

In this case, \(v^i_y\) cannot be private, since \(v^i_x \prec_{o_D} v^i_y\) implies that there exists an edge connecting \(v^i_y\) to a node belonging to another agent. Further, if \(v^i_y\) were private, Theorem 1 states agent \(i\) can reason over it independently of agent \(j\).

So, by definition, \(e_{xy}\) belongs to \(E^i_X\). Thus, agent \(i\) would be explicitly forced to block in line 8, until receiving edge updates \(w_{zy}, w_{yz}\forall v, s.t. e_{xz} \in E^i\), which can only occur after \(v^i_y\) has been reinstated, updates calculated by agent \(j\) in lines 9-12, and edge update sent to agent \(i\) in line 14.

Hence, all three cases present contradictions, implying that it is impossible for agent \(i\) to reinstate \(v^i_x\) prior to \(v^i_y\) when \(v^i_x \prec_{o_D} v^i_y\).
Conclusion: Hence we have shown that $\Delta\Delta$PPC either achieves the same output as applying $\Delta$DPC and $\Delta$PPC with respect to $\prec^\Delta$, and thus establishes PPC on $\mathcal{G}$ that is equivalent to $P^3C (o_D, \mathcal{G})$.

A.5 The MaTD Algorithm Proof of Completeness

Here we prove Theorem 10, originally stated on page 135, which establishes the completeness of the MaTD algorithm (Algorithm 9 on page 134).

Theorem 10. The MaTD algorithm is complete.

Proof. The basic intuition for this proof is provided by the fact that the MaTD algorithm is simply a more general, distributed version of the basic backtrack-free assignment procedure that can be consistently applied to a DPC distance graph. We show that when we choose bounds for new, unary decoupling constraints for $v_k$ (effectively in line 12), $w_{zk}, w_{kz}$ are path consistent with respect to all other variables. This is because not only is the distance graph DPC, but also the updates in lines 10-11 guarantee that $w_{zk}, w_{kz}$ are path consistent with respect to $v_j$ for all $j > k$ (since each such path from $v_j$ to $v_k$ will be represented as an edge $e_{jk}$ in the distance graph). So the only proactive edge tightening that occurs, which happens in line 12 and guarantees that $w_{zk} + w_{kz} = 0$, is done on path-consistent edges and thus will never introduce a negative cycle (or empty domain).

Fact 1: After lines 1-2 of the MaTD algorithm (Algorithm 9; page 134), if no decoupling exists, line 2 is guaranteed to terminate the algorithm by returning inconsistent, since, by definition, any consistent MaSTP has at least one solution schedule, which is a de facto temporal decoupling.

Fact 2: Lines 1-2 of the MaTD algorithm establish DPC, which implies that for every (external) timepoint variable $v_k$, the weights of all edges involving $v_k$ (including, in particular, the weights of $e_{zk}, w_{zk}$ and $w_{kz}$), are directionally path consistent with respect to all variables $v_j$ such that $v_j$ appears before $v_k$ in $o_S$.

Now we will show by induction for every external timepoint $v_k$ that the decoupling bounds computed in line 12 and constructed in line 14 are path consistent with respect to every other variable $v_j$ where $j > k$ in $o_S$ (Part 1) and form a non-empty domain (that is $b_{zk} + b_{kz} \geq 0$) (Part 2).

Base case ($k = n$): The base case is trivial, since when $k = n$ there exist no $v_j$ such that $j > k$. Thus upon entering line 12, $w_{zk}$ and $w_{kz}$ are path consistent with respect to every variable $v_j$ where $j \neq k$ (Fact 2). Also, since line 2 returns inconsistent if the problem instance is, this guarantees that $w_{zk} + w_{kz} \geq 0$ (Fact 1). In line 12, the incoming weights $w_{zk}$ and $w_{kz}$ are either left unchanged or tightened, but not beyond $w_{zk} + w_{kz} \geq 0$. Thus the bounds constructed in line 14 are path consistent.

Inductive case ($k < n$): Assume that the bounds of all decoupling constraints chosen for all variables $v_j$ for $j = k+1, \ldots, n$ are partially path consistent, that is $w_{jz} + w_{zj} \geq 0$, $w_{jz} \leq w_{jx} + w_{xz}$, and $w_{zj} \leq w_{xz} + w_{xj}$ for all $x \neq j$. 

8
Part 1: Here we show that the bounds of the decoupling constraints computed in line 12 and constructed in line 14 are as least as tight as the tightest existing path between \( v_k \) and \( z \). By contradiction, assume there exists some timepoint \( v_j \) where \( j > k \) in \( \sigma_S \) such that WLOG \( w_{kz} > w_{kj} + w_{jz} \). Note that since DPC is established in lines 1-2, any path from \( v_j \) to \( v_k \) will be represented as an edge \( e_{jk} \) with path consistent weights \( w_{jk}, w_{kj} \) in the distance graph (Fact 1). Notice also that if \( v_j \) is local to the agent of \( v_k \), then the update in line 12 ensures that \( w_{kz} \leq w_{kj} + w_{jz} \), thus \( v_j \) must be external to the agent of \( v_k \). However, then the update in line 10 ensures that \( w_{zk} \leq w_{jz} \). Since we inductively assumed that \( w_{jz} \) was chosen to be path consistent, \( w_{jz} \geq w_{jz} \). So the update in line 11 implies \( w_{kz} \leq w_{kj} - w_{zj} \leq w_{kj} + w_{jz} \). Thus there is a contradiction since we have shown that lines 10,12 (and for \( w_{zk} \), lines 11,12) ensure that \( w_{kz} \) and \( w_{zk} \) represent the tightest path between \( v_k \) and \( z \). This implies that line 9 introduces a contradiction since we have shown that lines 10,12 (and for \( w_{zk} \), lines 11,12) ensure that \( w_{kz} \) and \( w_{zk} \) represent the tightest path between \( v_k \) and \( z \). Thus the bounds chosen in line 14 are guaranteed to be at least as tight as any existing path between \( v_k \) and \( z \).

Part 2. Here we show that the bounds of the decoupling constraints constructed in line 14 form a non-empty domain. By contradiction, assume that \( w_{kz} + w_{zk} < 0 \). Once DPC is established in lines 1-2, (at which point INCONSISTENT is returned for any input distance graphs with negative cycles), \( w_{zk} \) and \( w_{kz} \) are tightened in lines 9,10-11, and 12. However, notice that line 12 guarantees that \( w_{zk} + w_{kz} > 0 \) and lines 10-11 simply recovers path consistency inductively. Thus, assume that line 9 introduces the negative cycle. That is, there exists some \( v_x \) and \( v_y \) such that \( w_{zk} = w_{zk} - w_{zx} \), and \( w_{kz} = w_{kz} - w_{xz} \) and where \( x, y > k \) in \( \sigma_S \) and \( v_x, v_y \) are external to the agent of \( v_k \), which together implies \( w_{zk} + w_{kz} < 0 \). Then, if \( x = y \),

\[
\begin{align*}
  w_{zk} - w_{xz} + w_{ky} - w_{zy} &< 0 \\
  \rightarrow w_{xz} + w_{zk} + w_{kx} + w_{xz} &< 0 \\
  \rightarrow w_{zk} + w_{kx} &< 0
\end{align*}
\]

where (2) holds by simple replacement \((x = y)\), and (3) holds inductively \((x = w_{xz} + w_{xx} = 0)\). However, (3) is a contradiction, since the only time \( e_{zk} \) will have been updated is during the DPC, which for this case would have returned INCONSISTENT.

So \( x \neq y \). WLOG, let \( v_x \) appear before \( v_y \) in \( \sigma_S \). Then,

\[
\begin{align*}
  w_{zk} - w_{xz} + w_{ky} - w_{zy} &< 0 \\
  \rightarrow w_{xz} + w_{zk} + w_{kx} + w_{x} &< 0 \\
  \rightarrow w_{xz} + w_{xy} + w_{y} &< 0 \\
  \rightarrow w_{xz} + w_{xz} &< 0
\end{align*}
\]

where (5) holds inductively \((x = w_{xz} + w_{xx} = 0; w_{yz} + w_{zy} = 0)\), (6) holds since DPC is established in lines 1-2 \((x = w_{xy} \leq w_{zk} + w_{ky})\), and (7) holds since \( x < y \) in \( \sigma_S \), and thus line 10 (depending on whether \( e_{xy} \) is external or not) ensures \( w_{xz} \leq w_{xy} - w_{zy} = w_{xy} + w_{yz} \). However, (7) is an obvious contradiction. Thus, the decoupling bounds chosen for \( v_k \) are guaranteed to form a non-empty domain.
Therefore we have shown inductively that the decoupling bounds chosen for $v_k$ are at least as tight as the tightest possible path between $v_k$ and $z$ and always form a non-empty domain. Thus, Algorithm 9 always finds a temporal decoupling of a MaSTN, if one exists.

A.6 The MaTDR Proof of Minimal Decoupling

Here we prove Theorem 12, originally stated on page 138, which establishes that the constraints that the MaTDR algorithm (Algorithm 10 on page 137) generates form a minimal temporal decoupling.

Theorem 12. The local constraints calculated by the MaTDR algorithm form a minimal temporal decoupling of $S$.

Proof. Notice, that the MaTDR subroutine is only called if the input network is consistent (and a valid decoupling has been found). We prove by contradiction that if any bound on an edge in $C'_\Delta$ is relaxed, $C'_\Delta$ may no longer form a temporal decoupling of $G$. Assume there exists a bound of an edge in $C'_\Delta$ that can be relaxed such that $C'_\Delta$ still forms a decoupling of $G$. WLOG, let $\delta_{xz}$ be the bound on edge $e_{xz}$ that can be relaxed by some positive value $\epsilon_{xz}$ and still form a temporal decoupling of $G$.

Notice that during the execution of the MaTDR, $\delta_{xz}$ is updated exclusively in line 9, and WLOG, let the loop where $j = y$ be the last time that $\delta_{xz}$ is updated, that is, $\delta_{xz} < w_{xy} - \delta_{zy}$. Then after line 9 is executed, $\delta_{xz} = w_{xy} - \delta_{zy}$.

If $v_y$ appears before $v_x$ in $o$, then $\delta_{zy}$ will have already been updated (prior to $\delta_{xz}$ due to line 8). But this leads to a contradiction, since $\delta_{xz} + \epsilon_{xz} + \delta_{zy} > w_{xy}$ since $\epsilon_{xz}$ is positive, and thus a bound of $\delta_{xz} + \epsilon_{xz}$ would no longer imply that $e_{xy}$ will be satisfied.

Thus, $v_y$ must appear after $v_x$ in $o$. Let $\delta^{IN}_{zy}$ and $\delta^{OUT}_{zy}$ be the input and output values of $\delta_{zy}$ respectively. Then, as already shown $\delta_{xz} + \delta_{zy}^{IN} = w_{xy}$ and by our assumption $\delta_{xz} + \epsilon_{xz} + \delta_{zy}^{OUT} \leq w_{xy}$, which implies $\delta_{zy}^{OUT} \leq \delta_{zy}^{IN} - \epsilon_{xz}$. For this to be true, there must exist some timepoint $v_w$ such that $w \neq x$, $w$ appears before $y$ in $o$, and $\delta_{zy}^{OUT} = w_{wy} - \delta_{wz}$. Then, $w_{wy} - \delta_{wz} \leq \delta_{zy}^{IN} - \epsilon_{xz}$. However, line 9 would have guaranteed that $\delta_{wz} \leq w_{wy} - \delta_{zy}^{IN}$ and so $\delta_{zy}^{IN} \leq w_{wy} - \delta_{wz}$, which leads to the contradiction $\delta_{zy}^{IN} \leq \delta_{zy}^{IN} - \epsilon_{xz}$.

Therefore, if any bound on any edge in $C'_\Delta$ is relaxed, $C'_\Delta$ may no longer by a decoupling of $G$. In other words, if we relaxed a bound of some edge in $C'_\Delta$, the bound on some other edge in $C'_\Delta$ must be tightened to guarantee that $C'_\Delta$ decouples $G$.  

\[ \]