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 D△DPC, and D△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 $\mathcal{G} = \langle V, E\rangle$ and a total ordering o over V—that is $\forall v_x, v_y \in V$ such that $x \neq y$ implies $(v_x \prec_o v_y \vee v_y \prec_o v_x)$—let $\mathcal{G}_o^{\triangle} = \left\langle V, E \cup E_o^{\triangle}\right\rangle$ be the graph that results from triangulating graph $\mathcal{G}$ by eliminating vertices in order o and adding fill edges $E_o^{\triangle}$.*

Definition 2. *Given a graph $\mathcal{G} = \langle V, E\rangle$ and (total) elimination order o, we define the precedence relation $\prec_{\mathbf{o}}^{\triangle}$, where $v_x \prec_o^{\triangle} v_y$ 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, $v_x \prec_o^{\triangle} v_y \Leftrightarrow (v_x \prec_o v_y) \wedge (e_{xy}, e_{yx} \in E \cup E_o^{\triangle})$.*

Next we label the two key algorithmic operations of the DPC and PPC algorithms, ELIMINATE and REINSTATE respectively.

Definition 3. *We label lines 3-11 of the △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 △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 $\mathcal{G} = \langle V, E\rangle$. If o' is consistent with the precedence relation $\prec_o^{\triangle}$, then $\mathcal{G}_o^{\triangle} = \mathcal{G}_{o'}^{\triangle}$.*

Proof. Assume $\mathcal{G}_o^{\triangle} \neq \mathcal{G}_{o'}^{\triangle}$.

Since $\mathcal{G}_o^{\triangle} = \left\langle V, E \cup E_o^{\triangle}\right\rangle$ and $\mathcal{G}_{o'}^{\triangle} = \left\langle V, E \cup E_{o'}^{\triangle}\right\rangle$, if $\mathcal{G}_o^{\triangle} \neq \mathcal{G}_{o'}^{\triangle}$ then $E_o^{\triangle} \neq E_{o'}^{\triangle}$.

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

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 $E_{o'}^{\triangle}$ 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_{yz} respectively) at the time of its elimination. By

definition, this implies $v_z \prec_o^{\triangle} v_x$ and $v_z \prec_o^{\triangle} v_y$. So, under the assumption that o' respects the precedence relation $\prec_o^{\triangle}$, o' eliminates v_z prior to v_x and 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 $\mathcal{G}$, which implies it is a member of $E_o^{\triangle}$. 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 $E_o^{\triangle}$ under elimination order o that is not added to $E_{o'}^{\triangle}$ under o', *or* o' does *not* respect the precedence relation $\prec_o^{\triangle}$, but both cases violate the assumptions. Therefore, since every time elimination order o adds a fill edge e, it induces a new $\prec_o^{\triangle}$ relation, any other elimination order o' that also satisfies the relation $\prec_o^{\triangle}$ will also add the fill edge, implying $E_o^{\triangle} \subseteq E_{o'}^{\triangle}$.

Next we prove that $E_{o'}^{\triangle} \subseteq E_o^{\triangle}$, which mirrors the proof that $E_o^{\triangle} \subseteq E_{o'}^{\triangle}$. 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 $E_o^{\triangle}$ 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_{yz} 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 $E_o^{\triangle}$ 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 $v_x \prec_o^{\triangle} v_z$. This contradicts the assumption that o' respects $\prec_o^{\triangle}$. Therefore, since the order that vertices that share edges are specified as part of $\prec_o^{\triangle}$ by definition, if elimination order o' respects $\prec_o^{\triangle}$, $E_{o'}^{\triangle} \subseteq E_o^{\triangle}$

Since $E_o^{\triangle} \subseteq E_{o'}^{\triangle}$ and $E_{o'}^{\triangle} \subseteq E_o^{\triangle}$, $E_o^{\triangle} = E_{o'}^{\triangle}$ which violates the assumption that $\mathcal{G}_o^{\triangle} \neq \mathcal{G}_{o'}^{\triangle}$ since they only can differ in fill edges. Therefore, if o' is consistent with the precedence relation $\prec_o^{\triangle}$, then $\mathcal{G}_o^{\triangle} = \mathcal{G}_{o'}^{\triangle}$. □

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

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

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

Since $\triangle$DPC eliminates nodes with respect to the precedence relation $\prec_o^{\triangle}$, $\triangle$DPC *must* eliminate v_z before eliminating v_x and v_y, resulting in the update: $w_{xy}^{\triangle DPC} \leftarrow min(w_{xy}^{\triangle DPC}, w_{xz}^{\triangle DPC} + w_{zy}^{\triangle DPC})$, so unless the assumption that $\triangle$DPC respects $\prec_o^{\triangle}$ is violated, $\triangle$DPC correctly applies the update.

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

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

Since $w_{xy}^{\triangle DPC}$ is the *first* bound that $\triangle$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_{zy}^{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 $w_{xz}^{DPC} = w_{xz}^{\triangle DPC}$ and $w_{zy}^{DPC} = w_{zy}^{\triangle DPC}$, which is exactly the same update as $\triangle$DPC. Thus, $w_{xy}^{\triangle DPC} < w_{xy}^{DPC}$, 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 $v_x \prec_o^{\triangle} v_z$. This contradicts the assumption that the order $\triangle$DPC eliminates vertices respects $\prec_o^{\triangle}$. Therefore, $w_{xy}^{\triangle DPC} \geq w_{xy}^{DPC}$.

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

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

Proof. **Note:** When a vertex v_x is reinstated, both $\triangle$PPC and P^{3}C apply the following updates:

- $w_{xi} \leftarrow \min(w_{xi}, w_{xj} + w_{ji})$
- $w_{xj} \leftarrow \min(w_{xj}, w_{xi} + w_{ij})$
- $w_{ix} \leftarrow \min(w_{ix}, w_{ij} + w_{jx})$
- $w_{jx} \leftarrow \min(w_{jx}, w_{ji} + w_{ix})$

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

By contradiction: Assume that applying P^{3}C $(o, \Delta(o, \mathcal{G}'))$ achieves a different output than an application of $\triangle$PPC to $\mathcal{G}'$ that reinstates vertices in reverse $\prec_o^{\triangle}$ 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 $w_{xi}^{P^3C} \neq w_{xi}^{\triangle PPC}$. So either $w_{xi}^{P^3C} < w_{xi}^{\triangle PPC}$ or $w_{xi}^{P^3C} > w_{xi}^{\triangle PPC}$. WLOG, let $w_{xi}^{P^3C} \neq w_{xi}^{\triangle PPC}$ be the first such difference between $\mathcal{G}^{P^3C}$ and $\mathcal{G}^{\triangle PPC}$.

Part 1: Assume that after both P^{3}C and $\triangle$PPC are applied, $w_{xi}^{P^3C} < w_{xi}^{\triangle PPC}$.

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

Notice that the only time a bound $w_{ij}^{P^3C}$ is updated during P^{3}C 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} and e_{ij} exist in $\mathcal{G}'$, and v_x appears before v_i and v_j in o, then by definition, v_x will appear before v_i and v_j in $\prec_o^{\triangle}$, thus $\triangle$PPC will also apply this update. Since we assumed this was the first time P^{3}C and $\triangle$PPC differed, neither $w_{xj}^{P^3C} < w_{xj}^{\triangle PPC}$ nor $w_{ij}^{P^3C} < w_{ij}^{\triangle PPC}$ can be true.

Thus, there is a contradiction, and so $w_{xi}^{P^3C} \geq w_{xi}^{\triangle PPC}$.

Part 2: Assume that after both P^{3}C and $\triangle$PPC are applied, $w_{xi}^{\triangle PPC} < w_{xi}^{P^3C}$.

Since w_{xi} is the *first* place that P^{3}C applies a different update than $\triangle$PPC, the difference cannot occur as a result of a tighter bound $w_{xj}^{\triangle PPC} < w_{xj}^{P^3C}$ or $w_{ij}^{\triangle PPC} < w_{ij}^{P^3C}$ at the time of the update $w_{xi} \leftarrow \min(w_{xi}, w_{xj} + w_{ji})$. Thus, $\triangle$PPC must apply an update that P^{3}C 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 $\triangle$PPC's execution but not P^{3}C. However, this violates Lemma 2.

Case 2: $\triangle$PPC 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 $v_x \prec_o^{\triangle} v_k$, which violates the assumption that $\triangle$PPC reinstates vertices in reverse $\prec_o^{\triangle}$.

Therefore $w_{xi}^{P^3C} \leq w_{xi}^{\triangle PPC}$.

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

So far, we have defined a key precedence relation of graph triangulations, $\prec_o^{\triangle}$. We have shown that any elimination order o' that respects this precedence relation will result in the same triangulated graph. Further, we have shown that any application of the $\triangle$PPC

($\triangle$DPC) algorithm that respects the precedence relation $\prec_o^{\triangle}$ as it eliminates and reinstates vertices and tightens bounds will calculate exactly the same PPC (DPC) STN as applying the P^{3}C (DPC) algorithm using o. Notice that since we proved this for each phase of the P^{3}C algorithm independently, as long both phases of $\triangle$PPC respect $\prec_o^{\triangle}$, the total order in which it reinstates vertices in the two phases can be *different*.

We must now prove that both our D$\triangle$PPC and our D$\triangle$PPC algorithms correctly apply $\triangle$DPC and $\triangle$PPC respectively to calculate a PPC STP instance.

A.2 The D$\triangle$DPC Algorithm Proof of Correctness

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

Theorem 1. *D$\triangle$DPC correctly establishes DPC on the multiagent STP.*

Proof. Notice that the semantics of D$\triangle$DPC dictate that each agent i eliminates its private timepoints V_P^i in some order o_P^i before eliminating its shared timepoints V_S^i, 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 $o_P^i \forall i$ and o_S. WLOG, let $o_D = o_P^1 \wedge o_P^2 \wedge \cdots \wedge o_P^n \wedge o_S$, where $\wedge$ appends two orderings together. This proof proceeds to show that D$\triangle$DPC establishes DPC on $\mathcal{G}$ by showing that it calculates the same result as DPC$(o_D, \mathcal{G})$ by demonstrating that D$\triangle$DPC correctly applies DPC with respect to $\prec_{o_D}^{\triangle}$.

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

Assume that there exists some v_y such that $v_y \prec_{o_D}^{\triangle} v_x^i$ but has not been eliminated by the time agent i eliminates v_x^i.

Case 1: v_x^i and v_y belong to the same agent i.

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

Case 2: v_x^i is private and v_y belongs to some agent j where $i \neq j$.

Since v_x^i is private, by definition there can be no edge $e_{xy} \in E$. Further, since v_x^i 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_x^i 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_x^i is private, at the time that agent i executing D$\triangle$PPC eliminates v_x^i there can exist no v_y

such that $v_y \prec^{\triangle}_{o_D} v^i_x$ 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 $v_y \prec^{\triangle}_{o_D} v^i_x$ 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 $\prec^{\triangle}_{o_D}$.

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 $\prec^{\triangle}_{o_D}$, and so by Lemma 2, calculates the same output as DPC$(o_D, \mathcal{G})$.

□

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 $v^j_x \in V^i_X \cap V^j_L$ for which agent i is waiting to receive updated edges from agent j, while there is also a $v^i_y \in V^j_X \cap V^i_L$ 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. □

A.4 The D△PPC Algorithm Proof of Correctness

Here we prove Theorem 6, originally stated on page 124, which establishes the correctness of the D△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△PPC correctly establishes PPC on the MaSTN.*

Proof. Notice that the semantics of D△DPC dictate that each agent i eliminates its private timepoints V_P^i in some order o_P^i before eliminating its shared timepoints V_S^i, 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 $o_P^i \forall i$ and o_S. WLOG, let $o_D = o_P^1 \wedge o_P^2 \wedge \cdots \wedge o_P^n \wedge o_S$, where $\wedge$ appends two orderings together. This proof proceeds to show that D△PPC establishes PPC $\mathcal{G}$ by showing that it calculates the same result as P^{3}C $(o_D, \mathcal{G})$ by demonstrating that D△PPC reinstates vertices (and so correctly applies △PPC) with respect to $\prec_{o_D}^{\triangle}$.

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 △PPC algorithm reinstates v_x^i after it reinstates v_y if $v_x \prec_{o_D}^{\triangle} v_y$, it achieves the same out as P^{3}C $(o, \mathcal{G})$. We now show that, despite its concurrent execution, the last time D△PPC reinstates v_x is after it reinstates v_y if $v_x \prec_{o_D}^{\triangle} v_y$.

By contradiction, assume agent i reinstates v_x^i (i.e., applies line 3-14 of the D△PPC Algorithm) *before* v_y, despite the fact that $v_x^i \prec_{o_D}^{\triangle} v_y$.

Case 1: v_x^i and v_y belong to the same agent i.

Notice $v_x^i \prec_{o_D}^{\triangle} v_y$ implies $v_x^i \prec_{o_D} v_y$. However, if both v_y and v_x^i belong to agent i, they must both appear in o^i (i.e., $o_P^i \wedge o_S$) and therefore, by construction of o_D, $v_x^i \prec_{o^i} v_y$. However, line 3 explicitly reinstates nodes in reverse o^i order. Therefore, v_y must belong to some agent j where $i \neq j$.

Case 2: $v_y \in V_P^j$ for some agent $j \neq i$.

$v_x^i \prec_{o_D}^{\triangle} v_y$ implies there is an edge between v_x^i and v_y at the time v_x^i is eliminated, which, by definition, implies v_y is not private. Further, if v_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_y cannot be private to some other agent j.

Case 3: $v_y \in V_X^i$, that is $v_y \in V_L^i$ for some agent $j \neq i$.

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

So, by definition, e_{xy} belongs to E_X^i. Thus, agent i would be explicitly forced to block in line 8, until receiving edge updates $w_{zy}, w_{yz} \forall v_z s.t. e_{xz} \in E^i$, which can only occur *after* v_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_x^i prior to v_y when $v_x^i \prec_{o_D}^{\triangle} v_y$.,

Conclusion: Hence we have shown that D△PPC either achieves the same output as applying △DPC and △PPC with respect to $\prec_o^{\triangle}$, and thus establishes PPC on $\mathcal{G}$ that is equivalent to P^{3}C $(o_D, \mathcal{G})$. □

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_k 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 $w_{zj} \leq w_{zx} + w_{xj}$ for all $x \neq j$.

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 o_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 $w_{zk} \leq w_{jk} - w_{jz}$. Since we inductively assumed that w_{jz} and w_{zj} were chosen to be path consistent, $w_{zj} \geq -w_{jz}$. 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 coming into line 12, which only further tightens w_{kz} and w_{zk} (if at all). 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 $w_{zk} + w_{kz} \geq 0$ and lines 10-11 simply recovers path consistency with respect to any local variable v_j where $j > k$ in o_S, which is guaranteed to be path consistent based on the inductive assumption. This implies that line 9 introduces the negative cycle. That is, there exists some v_x and v_y such that $w_{zk} = w_{xk} - w_{xz}$ and $w_{kz} = w_{kx} - w_{zx}$ and where $x, y > k$ in o_S and v_x,v_y are external to the agent of v_k, which together implies $w_{xk} - w_{xz} + w_{ky} - w_{zy} < 0$. Then, if $x = y$,

$$
\begin{aligned}
& w_{xk} - w_{xz} + w_{ky} - w_{zy} < 0 && (1)\\
\rightarrow \quad & w_{zx} + w_{xk} + w_{kx} + w_{xz} < 0 && (2)\\
\rightarrow \quad & w_{xk} + w_{kx} < 0 && (3)
\end{aligned}
$$

where (2) holds by simple replacement ($x = y$), and (3) holds inductively (since $w_{xz} + w_{zx} = 0$). However, (3) is a contradiction, since the only time e_{xk} 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 o_S. Then,

$$
\begin{aligned}
& w_{xk} - w_{xz} + w_{ky} - w_{zy} < 0 && (4)\\
\rightarrow \quad & w_{zx} + w_{xk} + w_{ky} + w_{yz} < 0 && (5)\\
\rightarrow \quad & w_{zx} + w_{xy} + w_{yz} < 0 && (6)\\
\rightarrow \quad & w_{zx} + w_{xz} < 0 && (7)
\end{aligned}
$$

where (5) holds inductively (since $w_{xz} + w_{zx} = 0; w_{yz} + w_{zy} = 0$), (6) holds since DPC is established in lines 1-2 (since $w_{xy} \leq w_{xk} + w_{ky}$), and (7) holds since $x < y$ in o_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* $\mathcal{S}$.

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'_Δ is relaxed, C'_Δ may no longer form a temporal decoupling of $\mathcal{G}$. Assume there exists a bound of an edge in C'_Δ that can be relaxed such that C'_Δ still forms a decoupling of $\mathcal{G}$. WLOG, let δ_{xz} be the bound on edge e_{xz} that can be relaxed by some positive value ϵ_{xz} and still form a temporal decoupling of $\mathcal{G}$.

Notice that during the execution of the MaTDR, δ_{xz} is updated exclusively in line 9, and WLOG, let the loop where $j = y$ be the last time that δ_{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 δ_{zy} will have already been updated (prior to δ_{xz} due to line 8). But this leads to a contradiction, since $\delta_{xz} + \delta_{zy} = w_{xy}$ implies that $\delta_{xz} + \epsilon_{xz} + \delta_{zy} > w_{xy}$ since ϵ_{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 δ^{IN}_{zy} and δ^{OUT}_{zy} be the input and output values of δ_{zy} respectively. Then, as already shown $\delta_{xz} + \delta^{IN}_{zy} = w_{xy}$ and by our assumption $\delta_{xz} + \epsilon_{xz} + \delta^{OUT}_{zy} \leq w_{xy}$, which implies $\delta^{OUT}_{zy} \leq \delta^{IN}_{zy} - \epsilon_{xz}$. For this to be true, there must exist some timepoint v_w such that $w \neq x$, w appears before y in o, and $\delta^{OUT}_{zy} = w_{wy} - \delta_{wz}$. Then, $w_{wy} - \delta_{wz} \leq \delta^{IN}_{zy} - \epsilon_{xz}$. However, line 9 would have guaranteed that $\delta_{wz} \leq w_{wy} - \delta^{IN}_{zy}$ and so $\delta^{IN}_{zy} \leq w_{wy} - \delta_{wz}$, which leads to the contradiction $\delta^{IN}_{zy} \leq \delta^{IN}_{zy} - \epsilon_{xz}$.

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