The Byzantine Brides Problem

Swan Dubois* Sébastien Tixeuil† Nini Zhu‡

Abstract

We investigate the hardness of establishing as many stable marriages (that is, marriages that last forever) in a population whose memory is placed in some arbitrary state with respect to the considered problem, and where traitors try to jeopardize the whole process by behaving in a harmful manner. On the negative side, we demonstrate that no solution that is completely insensitive to traitors can exist, and we propose a protocol for the problem that is optimal with respect to the traitor containment radius.

1 Introduction

After 1123 years of existence, the Byzantine Empire finally collapsed soon after the fall of Constantinople in 1453 by the Ottoman army (see Figure 1). The various wars that opposed armies in the previous years ravaged their homeland as well as the capital city, as a contemporary reported [1]: "The blood flowed in the city like rainwater in the gutters after a sudden storm."

Figure 1: Scene from the battle defending Constantinople, Paris 1499

Allegedly, the main reason for the Byzantine defeat is that there were traitors amongst its leading generals [18, 13]. With traitors at their cores, armies suffered significant losses, leaving mostly widows, orphans, and devastated homes. After the country was taken and the truce signed, the city was to rebuild, starting with its core roots: families. In the ancient days, strict guidelines were followed to form new marriages, like coming from the same social circles or being of opposite

*UPMC Sorbonne Universités & Inria, France, swan.dubois@lip6.fr
†UPMC Sorbonne Universités & Institut Universitaire de France, France, sebastien.tixeuil@lip6.fr
‡UPMC Sorbonne Universités, France

sex. In a wasted land with few homes still standing, those were no longer sustainable options. Stability of marriages was decided to be the most important criterium, rendering every other consideration irrelevant. So, general guidelines were to be followed by all survivors: *(i)* do your best to make your marriage last, *(ii)* don't be picky about whom you are married to, and *(iii)* don't make others' marriage fail. Still, the Byzantine traitors that led the armies to their doom were hidden amongst the surviving population, and managed somehow to remain unnoticed. Their purpose was to cause as much havoc as possible, by any means necessary, without being caught for their socially inconvenient behavior. So, the reconstruction of the city could have been jeopardized by few nasty Byzantine brides or bridegrooms.

The core problem Byzantine authorities were facing to establish as many stable marriages as possible lied in the following two observations:

1. the population was heavily shocked by the war that just stopped, and their state of mind was somewhat erratic: some could not remember they were previously married, some though they were previously married but never were, some though they were engaged and expected a response that would never come because the engagement was not remembered by the expected bride or bridegroom, etc,

2. the traitors could simulate emotional shock in order to stay undiscovered yet try to perturbate the global marriage process.

So, the only difference between the general population and the traitors was their willingness to accommodate the stable marriage doctrine in their daily life.

In this paper, we investigate the hardness of establishing as many stable marriages (that is, marriages that last foverer) in a population whose memory is placed in some arbitrary state with respect to the considered problem, and where traitors try to jeopardize the whole process by behaving in a harmuful manner. On the negative side, we demonstrate that no solution that is completely insensitive to traitors can exist, and we propose a protocol for the problem that is optimal with respect to the traitor containment radius.

2 Model and Definitions

2.1 State Model

A *Byzantine city* $S = (V, L)$ consists of a set $V = \{v_1, v_2, \ldots, v_n\}$ of potential brides[1] (or simply brides) and a set L of potential marriages. A potential marriage is an unordered pair of distinct potential brides (this takes place before the Internet ages, so long distance marriage is not supposed to last forever, and only marriages occurring in a vicinity may be stable). A Byzantine city S can be regarded as a graph whose vertex set is V and whose link set is L, so in the sequel we use graph terminology to describe a Byzantine city S. We use the following notations: $n = |V|$, $m = |L|$ and $d(u, v)$ denotes the distance between two nodes u and v (*i.e* the length of the shortest path between u and v).

Potential brides u and v are called *neighbors* if $(u, v) \in L$. The set of neighbors of a potential bride v is denoted by N_v. We do not assume existence of unique identifiers for potential brides (Birth records have been destroyed by the war, and memory of each potential bride is unreliable).

[1]Note that we use the word "bride" in the sequel of this paper to denote both brides and bridegrooms.

Instead we assume each potential bride may distinguish its neighbors from each other by locally labeling them.

For the sake of generality and the lack of reports concerning the remains of Constantinople after it has fallen, we consider that the Byzantine city has arbitrary yet connected topology. We adopt the *shared state model* [4] as a communication model, where each potential bride can directly and instantaneously get the current status of its neighbors.

The current memory that is maintained by a potential bride is denoted by the term of state, and may be further divided into variables. A potential bride may take actions that are prescribed by the authorities during the reconstruction of the Byzantine city. An action is simply a function that is executed in an atomic manner by the potential bride. The action executed by each potential bride is described by a finite set of guarded commands of the form $\langle \text{guard} \rangle \longrightarrow \langle \text{statement} \rangle$. Each guard of potential bride u is a Boolean expression involving the state of u and its neighbors.

A global state of a Byzantine city is called a *configuration* and is specified by the product of states of all potential brides. We define C to be the set of all possible configurations of a Byzantine city S. For a potential bride set $R \subseteq V$ and two configurations γ and γ', we denote $\gamma \overset{R}{\mapsto} \gamma'$ when γ changes to γ' by executing an action of each potential bride in R simultaneously. Notice that γ and γ' can be different only in the states of potential brides in R. For completeness of execution semantics, we should clarify the configuration resulting from simultaneous actions of neighboring potential brides. The action of a potential bride depends only on the current state at γ and the states of the neighbors at γ, and the result of the action reflects on the state of the potential bride at γ'.

We say that a potential bride is *enabled* in a configuration γ if the guard of at least one of its actions evaluates as true in γ. A *schedule* of a Byzantine city is an infinite sequence of potential bride sets. Let $Q = R^1, R^2, \ldots$ be a schedule, where $R^i \subseteq V$ holds for each i ($i \geq 1$). An infinite sequence of configurations $e = \gamma_0, \gamma_1, \ldots$ is called an *execution* from an initial configuration γ_0 by a schedule Q, if e satisfies $\gamma_{i-1} \overset{R^i}{\mapsto} \gamma_i$ for each i ($i \geq 1$). Potential bride actions are executed atomically, and we distinguish some properties on the scheduler (or daemon). A *distributed daemon* schedules the actions of potential brides such that any subset of potential brides can simultaneously execute their actions. We say that the daemon is *central* if it schedules action of only one potential bride at any step. The set of all possible executions from $\gamma_0 \in C$ is denoted by E_{γ_0}. The set of all possible executions is denoted by E, that is, $E = \bigcup_{\gamma \in C} E_\gamma$. We consider *asynchronous* Byzantine cities but we add the following assumption on schedules: any schedule is central and fair (meaning that only one enabled potential bride is chosen at any step and that no potential bride can be infinitely often enabled without being chosen by the scheduler)

In this paper, we consider (permanent) *Byzantine faults*: a Byzantine potential bride (*i.e.* a Byzantine-faulty potential bride) can exhibit arbitrary behavior independently of its actions. If v is a Byzantine-faulty potential bride, v can repeatedly change his (or her) state arbitrarily. For a given execution, the number of faulty potential brides is arbitrary.

2.2 Self-Stabilizing Protocols Resilient to Byzantine Faults

As the problem we solve is meant for stability and should reach a global fixed point, we use a *specification predicate* (shortly, specification) to define it. This specification predicate is denoted by $spec(v)$, for each potential bride v. A configuration is a desired one if every potential bride satisfies $spec(v)$. A specification $spec(v)$ is a Boolean expression on variables of P_v ($\subseteq P$) where P_v

is the set of potential brides whose state (or part of) appear in $spec(v)$. The variables appearing in the specification are called *output variables* (shortly, *O-variables*).

A *self-stabilizing protocol* ([4, 5, 20]) is a protocol that eventually reaches a *legitimate configuration*, where $spec(v)$ holds at every potential bride v, regardless of the initial configuration. Once it reaches a legitimate configuration, every potential bride never changes its O-variables and always satisfies $spec(v)$. From this definition, a self-stabilizing protocol is expected to recover from any number and any type of transient faults. However, the recovery from any configuration is guaranteed only when every potential bride honestly executes its action from the configuration, *i.e.*, self-stabilization does not consider the possibility of Byzantine-faulty potential brides.

When (permanent) Byzantine-faulty potential brides exist, they may not satisfy $spec(v)$. In addition, honest potential brides near the Byzantine-faulty potential brides can be influenced and may be unable to satisfy $spec(v)$. Nesterenko and Arora [17] define a *strictly stabilizing protocol* as a self-stabilizing protocol resilient to unbounded number of Byzantine-faulty actors.

Definition 1 *(c-honest potential bride) A potential bride is c-honest if it is honest (*i.e. *not Byzantine-faulty) and located at distance more than c from any Byzantine-faulty potential bride.*

Definition 2 *((c, f)-containment) A configuration γ is* (c, f)-contained *for specification spec if, given at most f Byzantine-faulty potential brides, in any execution starting from γ, every c-honest potential bride v always satisfies $spec(v)$ and never changes its O-variables.*

The parameter c of Definition 2 refers to the *containment radius* defined by Nesterenko and Arora [17]. The parameter f refers explicitly to the number of Byzantine-faulty potential brides, while [17] dealt with an arbitrary number of Byzantine faults (that is, $f \in \{0 \dots n\}$).

Definition 3 *((c, f)-strict stabilization) A protocol is* (c, f)-strictly stabilizing *for specification spec if, given at most f Byzantine-faulty potential brides, any execution $e = \gamma_0, \gamma_1, \dots$ contains a configuration γ_i that is (c, f)-contained for spec.*

A specification is r-restrictive [17] if it prevents combinations of states that belong to two potential brides u and v that are at least r hops away. An important consequence for our purpose is that the containment radius of protocols solving r-restrictive specifications is at least r.

3 Specification

The problem of maximal marriage construction is a well known problem in Distributed Computing. Given a graph $G = (V, E)$, a marriage M on G is a subset of E such that any node of V belongs to at most one edge of M. A marriage is maximal if there exists no marriage M' such that $M \subsetneq M'$.

Specification 1 *(Maximal Marriage)*

Liveness: The protocol terminates in a finite time.

Safety: In the terminal configuration, there exists a maximal marriage

Each potential bride v has a variable $pref_v$ which belongs to the set $N_v \cup \{null\}$. This variable refers to the preferred neighbor of v for a marriage. For example, if $pref_v = u$ then v wants to add the edge $\{v, u\}$ to the marriage. For any potential bride v, we define the following set of predicates over the Byzantine city: *(i)* $proposing_v$ denotes the fact that v is proposing marriage to some neighbor u, but that u has not shown interest yet, *(ii)* $married_v$ denotes that v has proposed u and u has proposed v back, *(iii)* $doomed_v$ denotes that v has proposed neighbor u, but u has proposed somebody else than v, *(iv)* $dead_v$ denotes that v has no hope of getting married ever (all neighbors proposed to somebody else), and *(v)* $single_v$ means that v has not proposed anyone and has at least one neighbor likewise. Formally:

$$\begin{array}{rcc} proposing_v & \equiv & (pref_v = u) \wedge (pref_u = null) \\ married_v & \equiv & (pref_v = u) \wedge (pref_u = v) \\ doomed_v & \equiv & (pref_v = u) \wedge (pref_u = w) \wedge (w \neq v) \\ dead_v & \equiv & (pref_v = null) \wedge (\forall u \in N_v, married(u) = true) \\ single_v & \equiv & (pref_v = null) \wedge (\exists u \in N_v, married(u) \neq true) \end{array}$$

It is easy to verify that for any configuration γ and for any potential bride v, exactly one of these predicates holds for v in γ.

If the Byzantine city is subject to Byzantine failures, obviously no protocol can satisfy the classical specification of the problem. Now, a potential bride v is considered locally legitimate when it satisfies the following predicate: $spec(v) \equiv married_v \vee dead_v$. We now describe the global properties that are satisfied by a (c, f)-contained configuration for *spec*. Informally, we can prove that there exists a maximal marriage on a subset of S in such a configuration and that this subset includes at least the set of c-honest potential brides. In the following, V_c denotes the set of c-honest potential brides (*i.e.*, $V_c = \{v \in V | \forall b \in B, d(v, b) > c\}$).

Definition 4 *((c, γ)-marriage subset) Given an integer $c > 0$ and a configuration γ, the (c, γ)-marriage subset $S^*_{c,\gamma}$ of S is the subset induced by the following set of potential brides:*

$$V' = V_c \cup \{v \in V \setminus V_c | \exists u \in V_c, pref_v = u \wedge pref_u = v\}$$

Now, we can state formally the property satisfied by any (c, f)-contained configuration for *spec*.

Lemma 1 *In any (c, f)-contained configuration for spec, there exists a maximal marriage on the subset $S^*_{c,\gamma}$.*

Proof Let γ be a (c, f)-contained configuration for *spec*. Hence, γ satisfies $\forall v \in V_c$, $married_v \vee dead_v$. Let us define the following edge set $M_c = \{\{v, pref_v\} | v \in V_c \wedge pref_v \neq null\}$.

First, we show that M_c is a marriage on $S^*_{c,\gamma}$. Indeed, if $\{v, pref_v\}$ is an edge of M_c, then v satisfies $married_v$ (since v satisfies $spec(v)$ and $pref_v \neq null$ by construction of M_c). Hence, we have $pref_{pref_v} = v$. Consequently, v and $pref_v$ appear only once in M_c.

Now, we show that M_c is maximal. By contradiction, assume it is not the case. Consequently, there exists two neighbors v and u (with $v \in V'$ and $u \in V'$) such that $\{v, u\} \notin M_c$ and $M'_c = M_c \cup \{\{v, u\}\}$ is a marriage on $S^*_{c,\gamma}$. Let us study the following cases:

Case 1: $u \in V_c$ and $v \in V_c$.

If $married_v \wedge married_u$ holds, then $\{v, u\} \in M_c$ by construction that contradicts the hypothesis. If $dead_v \wedge dead_u$ holds, then we can deduce that $(pref_v = null) \wedge (married_u)$ (since

$dead_v$ holds), that contradicts $dead_u$. If $dead_v \wedge married_u$ (resp. $married_v \wedge dead_u$) holds, then $\{v, pref_v\} \in M_c$ with $pref_v \neq u$ (resp. $\{u, pref_u\} \in M_c$ with $pref_u \neq v$) and we can deduce that v (resp. u) appears in two distinct edges of M'_c. Then, M'_c is not a marriage that contradicts the hypothesis.

Case 2: $u \notin V_c$ and $v \notin V_c$.
According to the assumption, $\{u, v\} \notin M_c$. Since $v \in V' \setminus V_c \wedge u \in V' \setminus V_c$, we have $\{v, pref_v\} \in M_c$ with $pref_v \neq u \wedge pref_v \in V_c$ (resp. $\{u, pref_u\} \in M_c$ with $pref_u \neq v \wedge pref_u \in V_c$) and we can deduce that v (resp. u) appears in two distinct edges of M'_c. Then, M'_c is not a marriage that contradicts the hypothesis.

Case 3: $u \in V_c$ and $v \notin V_c$.
According to the assumption, $\{v, u\} \notin M_c$. Since $v \in V' \setminus V_c \wedge u \in V_c$,we have $\{v, pref_v\} \in M_c$ with $pref_v \neq u \wedge pref_v \in V_c$ (since if $pref_v = u$, then $\{v, u\} \in M_c$ that contradicts the hypothesis) and we can deduce that v appears in two distinct edges of M'_c. Then, M'_c is not a marriage that contradicts the hypothesis.

□

The result of Lemma 1 motivates the design of a strictly stabilizing protocol for *spec*. Indeed, even if this specification is local, it induces a global property in (c, f)-contained configuration for *spec* since there exists a maximal marriage of a well-defined sub-graph in such a configuration.

4 Strictly Stabilizing Maximal Marriage

This section presents our strictly stabilizing solution for the maximal marriage problem. We also prove its correctness and its optimality with respect to containment radius.

4.1 Our Protocol

Our strictly-stabilizing maximal marriage protocol, called $\mathcal{SSMM}$ is formally presented as Algorithm 1. The basis of the protocol is the well-known self-stabilizing Maximal Marriage protocol by Huang and Hsu [12], but we allow potential brides to remember their past sentimental failures (*e.g.* an aborted marriage du to the mate being Byzantine-faulty, or a proposal that didn't end up in an actual marriage) in order not to repeat the same mistakes forever when Byzantine-faulty brides participate to the global marriage process. The ideas that underly the marriage process for honest potential brides follows the directives discussed in the introduction: *(i)* once married, honest brides never divorce and never propose to anyone else, *(ii)* honest brides may propose to any neighbor, and if proposed, will accept marriage gratefully, *(iii)* if they realize they previously proposed to somebody that is potentially married to somebody else, they will cancel their proposal and refrain proposing to the same potential bride soon. A potential bride v maintain two variables: $pref_v$, that was already discussed in the problem specification section, and old_pref_v that is meant to recall past sentimental failures. Specifically, old_pref_v stores the last proposal made to a neighbor that ended up doomed (because that neighbor preferred somebody else, potentially because of Byzantine-faulty divorce, or because of genuine other interest that occurred concurrently). Then, the helper function $next\_v$ helps v to move on with past failures by preferring the next mate not to be the same as previously (in a Round Robin order): the same potential bride that caused a sentimental breakup may be chosen twice in a row only if the only one available.

algorithm 1 $\mathcal{SSMM}$: Strictly-stabilizing maximal marriage for potential bride v

Variables:

$pref_v \in N_v \cup \{null\}$: preferred neighbor of v
$old\_pref_v \in N_v$: previous preferred neighbor of v

Function:

For any $u \in \{v, null\}$, $next_v(u)$ is the first neighbor of v greater than old_pref_v (according to a round robin order) such that $pref_{next_v(u)} = u$

Rules:

/* Don't be picky: Accept any mate (round robin priority) */
$(M) :: (pref_v = null) \wedge (\exists u \in N_v, pref_u = v) \longrightarrow pref_v := next_v(v)$
/* Don't be picky: Propose to anyone (round robin priority) */
$(S) :: (pref_v = null) \wedge (\forall u \in N_v, pref_u \neq v) \wedge (\exists u \in N_v, pref_u = null) \longrightarrow pref_v := next_v(null)$
/* Don't cause others to break up: give up proposing if doomed */
$(A) :: (pref_v = u) \wedge (pref_u \neq v) \wedge (pref_u \neq null) \longrightarrow old\_pref_v := pref_v; pref_v := null$

4.2 Proof of Strict Stabilization

In their paper [12], Hsu and Huang prove the self-stabilizing property of their maximal marriage algorithm using a variant function. A variant function is a function that associates to any configuration a numerical value. This function is designed such that: *(i)* the function is bounded, *(ii)* any possible step of the algorithm decreases strictly the value of the function, and *(iii)* the function reaches its minimal value if and only if the corresponding configuration is legitimate. Once such a function is defined and its properties are proved, we can easily deduce the convergence of the protocol. Indeed, whatever the initial configuration is, the associate value by the variant function is bounded (by property *(i)*) and any execution starting from this configuration reaches in a finite time the minimal value of the function (by property *(ii)*). Then, property *(iii)* allows us to conclude on the convergence of the algorithm.

Our proof of strict-stabilization for our protocol also relies on a variant function (borrowed from the one of [19]). We choose a variant function where we consider only potential brides of V_2. For any configuration $\gamma \in \Gamma$, let us define the following functions:

$$\begin{aligned} w(\gamma) &= |\{v \in V_2 | proposing_v\}| \\ c(\gamma) &= |\{v \in V_2 | doomed_v\}| \\ f(\gamma) &= |\{v \in V_2 | single_v\}| \\ P(\gamma) &= (w(\gamma) + c(\gamma) + f(\gamma), 2c(\gamma) + f(\gamma)) \end{aligned}$$

Note that our variant function P satisfies property *(i)* by construction.

Then, we define the following configuration set:

$$\mathcal{LC}_2 = \{\gamma \in \Gamma | \forall v \in V_2, spec(v)\}$$

In other words, $\mathcal{LC}_2$ is the set of configurations in which any potential bride v of V_2 satisfies $spec(v)$.

We can now explain the road-map of our proof. After two preliminaries results (Lemmas 2 and 3) that are used in the sequel, we first show that any configuration of the set $\mathcal{LC}_2$ is $(2, n)$-contained

for *spec* (Lemma 4), that is, the set $\mathcal{LC}_2$ is closed by actions of $\mathcal{SSMM}$. Then, there remains to prove the convergence of the protocol to configurations of $\mathcal{LC}_2$ (starting from any configuration) to show the strict-stabilization of $\mathcal{SSMM}$. The remainder of the proof is devoted to the study of properties of our variant function P. First, we show in Lemma 5 that any configuration γ that satisfies $P(\gamma) = (0, 0)$ belongs to $\mathcal{LC}_2$. This proves that P satisfies the property *(iii)*. Unfortunately, we can prove that our variant function P does not satisfy property *(ii)* (strict decreasing) since Byzantine faults may lead some potential brides to take actions that increase the function value. Nevertheless, we prove in Lemmas 6, 7, and 8 that this case may appear only a finite number of times and that our variant function is eventually strictly decreasing, which is sufficient to prove the convergence to $\mathcal{LC}_2$ in Lemma 9. Finally, Lemmas 4 and 9 permit to conclude with Theorem 1 that establishes the $(2, n)$-strict stabilization of $\mathcal{SSMM}$. A detailed proof follows.

Lemma 2 *For any execution $e = \gamma_0, \gamma_1 \ldots$,*
- if $married_v$ holds in γ_0 for a potential bride $v \in V_1$, then $married_v$ holds in γ_i for all $i \in \mathbb{N}$; and
- if $dead_v$ holds in γ_0 for a potential bride $v \in V_2$, then $dead_v$ holds in γ_i for all $i \in \mathbb{N}$.

Proof Let v be a potential bride of V_1. Hence, any neighbor of v is a honest potential bride. If $married_v$ holds in a configuration γ_0, then $pref_v = u \wedge pref_u = v$ holds in γ_0 by definition. We can observe that v and u are not enabled by $(M), (S)$, or by (A) in γ_0. Consequently, v and u are not activated in any execution e starting from γ_0. In conclusion, $married_v$ holds in any configuration of e.

Let v be a potential bride of V_2. If $dead_v$ holds in a configuration γ_0, then $pref_v = null \wedge (\forall u \in N_v, married_u = true)$ holds in γ_0 by definition. Note that any neighbor of v belongs to V_1 (since $v \in V_2$). If $married_u$ holds in γ_0, then $married_u$ holds in any configuration of any execution starting from γ_0. Potential Bride v is not enabled by $\mathcal{SSMM}$ in γ_0. No neighbor of v is enabled (according to the first part of the proof). Consequently, $dead_v$ holds in any configuration of any execution starting from γ_0. □

Lemma 3 *For any configuration $\gamma \in \mathcal{LC}_2$, no potential bride of V_2 is enabled by $\mathcal{SSMM}$ in γ.*

Proof Let γ be a configuration of $\mathcal{LC}_2$. By definition, γ satisfies $\forall v \in V_2, married_v \vee dead_v$. Let v be a potential bride of V_2.

If $married_v$ holds in γ, then we have $pref_v = u$ and $pref_u = v$ by definition. We can observe that v is not enabled by rules (M) and (S) in γ since $pref \neq null$ and that v is not enabled by rule (A) in γ since $pref_u = v$.

If $dead_v$ holds in γ, then we have $pref_v = null$ and $\forall u \in N_v, married_u = true$ by definition. We can observe that v is not enabled by rule (A) in γ since $pref_v = null$ and that v is not enabled by rules (M) and (S) in γ since $\forall u \in N_v, married_u \Rightarrow \exists r_u, pref_u = r_u \neq v \neq null$.

In both case, v is not enabled in γ. Hence, no potential bride of V_2 is enabled in γ. □

The definition of $\mathcal{LC}_2$ and Lemma 2 allow us to state the following lemma:

Lemma 4 *Any configuration of $\mathcal{LC}_2$ is $(2, n)$-contained for spec.*

Lemma 5 *Any configuration $\gamma \in \Gamma$ satisfying $P(\gamma) = (0, 0)$ belongs to $\mathcal{LC}_2$.*

Proof If a configuration $\gamma \in \Gamma$ satisfies $P(\gamma) = (0, 0)$, then $w(\gamma) + f(\gamma) + c(\gamma) = 0$. Hence, no potential bride of V_2 is proposing, single, or doomed. Every potential bride v of V_2 satisfies $married_v \vee dead_v$. By definition of $\mathcal{LC}_2$, we have $\gamma \in \mathcal{LC}_2$. □

The following lemma is proved in a similar way as the corresponding one of [19] (considering only potential brides of V_2).

Lemma 6 *For any configuration $\gamma \notin \mathcal{LC}_2$ and any step $\gamma \to \gamma'$ in which a potential bride of V_2 is activated by $\mathcal{SSMM}$, we have $P(\gamma') < P(\gamma)$.*

Proof Let γ be a configuration such that $\gamma \notin \mathcal{LC}_2$. Consider any step $\gamma \to \gamma'$ of $\mathcal{SSMM}$. Since the scheduling is central, at most one potential bride $v \in V_2$ is activated during $\gamma \to \gamma'$. Consider the following cases.
Case 1: $v \in V_2$ is activated by rule (M) during $\gamma \to \gamma'$.
By construction, there exists $u \in N_v$ such that v and u become married during this step. Hence, the function $w+f+c$ decreases by at least 1 during this step. Consequently, we have $P(\gamma') < P(\gamma)$.
Case 2: $v \in V_2$ is activated by rule (S) during $\gamma \to \gamma'$.
As $pref_v = null$ and there exists $u \in N_v$ such that $pref_u = null$ in γ, $single_v$ holds in γ. On the other hand, $pref_v = u$ and $pref_u = null$ hold in γ', that implies that $proposing_v$ holds in γ'. Hence, the function $2c + f$ decreases at least by one during $\gamma \to \gamma'$.

As rule (S) is enabled in γ, we can deduce that no neighbor of v is proposing after it (otherwise, the rule (S) is not enabled for v). So no proposing node in γ becomes single or doomed in γ'. If a neighbor of v is single in γ, it remains single or become dead in γ'. We can conclude that the function $c + f + w$ remains equal and that the function $2c + f$ decreases by exactly one during $\gamma \to \gamma'$. Consequently, we have $P(\gamma') < P(\gamma)$.
Case 3: $v \in V_2$ is activated by rule (A) during $\gamma \to \gamma'$.
As there exists $u \in N_v$ such that $pref_v = u$, $pref_u = w$, and $w \neq v$ in γ, we can deduce that $doomed_v$ holds in γ. As $pref_v = null$ holds in γ', we know that $single_v \vee dead_v$ holds in γ'. Hence, the function $2c + f$ decreases by at least one during $\gamma \to \gamma'$.

If a neighbor of v is single in γ, then it remains single in γ' (note that u cannot become dead in γ' since $pref_v = null$). If a neighbor of v is proposing in γ, it remains in this state because it cannot wait for v (recall that $pref_v \neq null$ in γ). If a neighbor of v is doomed to v in γ, then it becomes proposing in γ'. Note that, if $u \in V_2$, u leads to a supplementary decreasing of the function $2c + f$ while $c + w + f$ remains equal but, if $u \notin V_2$, then functions $2c + f$ and $c + w + f$ remains equal. We can conclude that the function $2c + f$ decreases by at least one during $\gamma \to \gamma'$. Consequently, we have $P(\gamma') < P(\gamma)$. □

Lemma 7 *In any execution, P only increases a finite number of times.*

Proof Let v be a potential bride of V. Let e be an execution in which P is not monotonically decreasing. Consider the first step $\gamma \to \gamma'$ of e in which P increases and in which v is activated.

By Lemma 6, we know that $v \notin V_2$ (otherwise, we have a contradiction with the decreasing of P). Then, by construction of P, we know that $v \in V_1$ (since actions of potential brides of V_0 have no effects on values of P). Consequently, $v \in V_1 \setminus V_2$.

Assume that v executes rule (S) during the step $\gamma \to \gamma'$. Observe that v is *single* in γ but becomes *proposing* in γ. Moreover, any neighbor of v that is *single* in γ remains in this state in γ' and there is no neighbor *dead*, *proposing* after v, or *doomed* after v in γ (by construction of the rule). By Lemma 2, any neighbor of v in V_2 remains married in γ' if it is *married* in γ. Hence, the action of v does not modify the state of its neighbors and P is not modified, that contradicts the assumption.

Assume that v executes rule (A) during the step $\gamma \to \gamma'$. Observe that v is *doomed* in γ but becomes *single* in γ'. Moreover, any neighbor of v that is *single* in γ remains in this state in γ'. By construction of the rule, there is no neighbor of v that is *dead* or *proposing* after v in γ. Any neighbor of v that is *doomed* after v in γ becomes *proposing* in γ'. By Lemma 2, any neighbor of v in V_2 remains married in γ' if it is *married* in γ. Hence, the action of v leads to a strict decreasing of P, that contradicts the assumption.

Consequently, we know that v is activated by rule (M) during the step $\gamma \to \gamma'$. Then, by construction of the rule, we know that v becomes *married* in γ' and remains in this state during the whole execution (by Lemma 2). In particular, v is never activated in the sequel of the execution.

In conclusion, we obtain that each potential bride of $v \in V_1 \setminus V_2$ executes at most one step that decreases P. As the number of potential bride of $v \in V_1 \setminus V_2$ is finite, we obtain the result. □

Lemma 8 *For any configuration $\gamma_0 \notin \mathcal{LC}_2$ and any execution $e = \gamma_0, \gamma_1, \gamma_2, \ldots$ starting from γ_0, there exists a configuration γ_i such that $P(\gamma_{i+1}) < P(\gamma_i)$.*

Proof Let γ_0 be a configuration such that $\gamma_0 \notin \mathcal{LC}_2$. By contradiction, assume that there exists an execution $e = \gamma_0, \gamma_1, \gamma_2, \ldots$ starting from γ_0 such that for any $i \in \mathbb{N}$, $P(\gamma_{i+1}) \geq P(\gamma_i)$. By Lemma 6, that implies that no potential bride of V_2 is activated in any step of e.

As $\gamma_0 \notin \mathcal{LC}_2$, there exists $v \in V_2$ such that $spec(v)$ does not hold in γ_0. Hence, v is *proposing*, *doomed*, or *single* in γ_0. Consider the following cases.

Case 1: v is *proposing* in γ_0. By definition, we have $\exists u \in N_v, (pref_v = u) \wedge (pref_u = null)$.

Case 1.1: If $u \in V_2$, then we can observe that u is enabled by (M) in γ_0. Since v and u are never activated in e (by Lemma 6), then u remains continuously enabled. As the daemon is fair, u is activated in a finite time, that is contradictory.

Case 1.2: If $u \notin V_2$, then we can observe that u is continuously enabled by (M) from γ_0 (since v is never activated). As the daemon is fair, u executes (M) in a finite time and becomes *married*. If u is married with v, then P decreases, that is contradictory. We can deduce that u becomes *married* with another potential bride and is never activated afterwards (by Lemma 2). Then, v becomes *doomed* and continuously enabled by rule (A). As the daemon is fair, v is activated in a finite time and becomes *dead* or *single*. In both cases, P decreases, that is contradictory.

Case 2: v is *doomed* in γ_0. By definition, we have $(pref_v = u) \wedge (pref_u = r) \wedge (r \neq v)$.

Case 2.1: If u is activated in e, then we can observe that v is continuously enabled by rule (A). As the daemon is fair, v is activated in a finite time, that is contradictory.

Case 2.2: If u is activated in e, then we know that $u \notin V_2$ (otherwise, we obtain a contradiction). Before the first activation of u, we have $(pref_v = u) \wedge (pref_u = r) \wedge (pref_r = w) \wedge (r \neq v) \wedge (w \neq u)$ since the only enabled rule when $pref_u \neq null$ is (A). After the execution of (A) by u, v becomes *proposing* after u and we can refer to case 1.2.

Case 3: If $single_v$ holds in γ_0, then we have $(pref_v = null) \wedge (\exists u \in N_v, married_u = false)$ by definition. Let us study the following cases.

Case 3.1: $u \in V_2$.

By Lemma 6, we know that u is never activated in e. Consequently, the fairness of the daemon allows us to conclude that $pref_u = r \neq v$. Indeed, in the contrary case, v is continuously enabled by (M) if $pref_u = v$ and u and v are continuously enabled by (M) or by (S) if $pref_u = null$.

If u is *doomed*, we can refer to case 2 with u playing the role of v while if u is *proposing*, we can refer to case 1 with u playing the role of v, that ends this case.

Case 3.2: $u \in V_1 \setminus V_2$
Observe that u cannot be *dead* since v is *single*. If u is *single*, then u is continuously enabled by (S) or by (M). If u is *doomed*, then u is enabled by (A). If u is *proposing* after a potential bride r different than v, then r is continuously enabled by (S) or (M). The fairness of the daemon implies that r is activated in a finite time and hence that u remains *proposing* only a finite time.

Consequently, we know that, while u is not activated, remains not *married*, and is not *proposing* after v, u is infinitely often enabled. The fairness of the daemon implies that, while u is not *married* nor *proposing* after v, u is activated in a finite time. The construction of the algorithm and the round robin policy used for the management of the pointer ensure us that u is either *married* or *proposing* after v in a finite time.

If u becomes *proposing* after v, then v becomes enabled by (M) and u is not enabled while v is not activated. Hence, the fairness of the daemon leads to an activation of v in a finite time, that is contradictory.

If u becomes *married*, then v can remain *single* or become *dead*. The first case allows us to refer to case 3 again (but this case can arise only a finite number of times since the number of neighbors of v is finite). In the second case, we obtain a contradiction since P strictly decrease.

In any case, we obtain a contradiction in a finite time and we can deduce the lemma. □

This set of Lemmas allows us to conclude on the following results:

Lemma 9 *Any execution of $\mathcal{SSMM}$ reaches a configuration of $\mathcal{LC}_2$ in a finite time under the central fair daemon.*

Theorem 1 *$\mathcal{SSMM}$ is a $(2, n)$-strictly stabilizing protocol for spec under the central fair daemon.*

4.3 Optimality of Containment Radius

This section is devoted to the impossibility result that proves the optimality of the containment radius performed by $\mathcal{SSMM}$.

Theorem 2 *There exists no $(1, 1)$-strictly stabilizing protocol for spec under any daemon.*

Proof Consider a Byzantine city reduced to a chain of 5 potential brides labelled from left to right by v_0, v_1, ..., v_4. Consider the configuration γ in which v_0 (resp. v_3) is married with v_1 (resp. v_4). Hence, v_2 is dead. Observe that γ belongs to $\mathcal{LC}_1$ if the potential bride v_0 is Byzantine-faulty (*i.e* any potential bride of V_1 is either *married* or *dead*).

By definition, any $(1, 1)$-strictly stabilizing protocol for *spec* must ensure the closure of $\mathcal{LC}_1$ for any execution starting from γ. But we can observe that it is not the case. Indeed, it is sufficient that the Byzantine-faulty potential bride breaks its marriage with v_1 during the first step for violating the closure of $\mathcal{LC}_1$ (since $v_2 \in V_1$ becomes *single*). As no protocol can prevent a Byzantine fault by definition, we have the result. □

5 Related Works

Self-stabilization [4, 5, 20] is a versatile technique that permits forward recovery from any kind of transient faults, while Byzantine fault-tolerance [13] is traditionally used to mask the effect of a limited number of malicious faults. In the context of self-stabilization, the first algorithm for

computing a maximal marriage was given by Hsu and Huang [12]. Goddard et al. [10] later gave a synchronous self-stabilizing variant of Hsu and Huang's algorithm. Finally, Manne et al. [14] gave an algorithm for computing a maximal marriage under the distributed daemon. When it comes to improving the $\frac{1}{2}$-approximation induced by the maximal mariage property, Ghosh et al. [9] and Blair and Manne [2] presented a framework that can be used for computing a maximum mariage in a tree, while Goddard et al. [11] gave a self-stabilizing algorithm for computing a $\frac{2}{3}$-approximation in anonymous rings of length not divisible by three. Manne et al. later generalized this result to any arbitrary topology [15]. Note that contrary to our proposal, none of the aforementioned marriage construction algorithms can tolerate Byzantine behaviour.

Making distributed systems tolerant to both transient and malicious faults is appealing yet proved difficult [6, 3] as impossibility results are expected in many cases (even with complete communication topology and in a synchronous setting). A promising path towards multi-tolerance to both transient and Byzantine faults is Byzantine containment. For local tasks (*i.e.* tasks whose correctness can be checked locally, such as vertex coloring, link coloring, or dining philosophers), strict stabilization [17, 16] permits to contain the influence of malicious behavior to a fixed radius. This notion was further generalized for global tasks (such as spanning tree construction) using the notion of topology-aware strict stabilization [7, 8]. Our proposal is a strictly stabilizing maximal marriage protocol that has optimal containement radius.

6 Conclusion

We investigated the problem of recovering a catastrophic war by establishing long standing marriages, despite starting from an arbitrarily devastated state and having traitors trying make the global process fail. We presented evidence that no protocol can be completely resilient to traitors (as far as their influence containment is concerned), and designed and formally proved a protocol to solve the problem that is optimal in that respect. Further work is still needed for determining the global possible efficiency of the marriage process. It is known that in a scenario without traitors, a given maximal marriage [12, 14] is a factor 2 from the optimal (over all possible maximal marriages), yet more efficient solutions (with respect to the approximation ration) are possible [15]. Extending those works to Byzantine-faulty setting is a challenging further work.

References

[1] Nicolo Barbaro. *Diary of the Siege of Constantinople.* Translation by John Melville-Jones, New York, 1453.

[2] Jean R. S. Blair and Fredrik Manne. Efficient self-stabilizing algorithms for tree network. In *ICDCS*, pages 20–, 2003.

[3] Ariel Daliot and Danny Dolev. Self-stabilization of byzantine protocols. In Ted Herman and Sébastien Tixeuil, editors, *Self-Stabilizing Systems*, volume 3764 of *Lecture Notes in Computer Science*, pages 48–67. Springer, 2005.

[4] Edsger W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. ACM*, 17(11):643–644, 1974.

[5] Shlomi. Dolev. *Self-stabilization.* MIT Press, March 2000.

[6] Shlomi Dolev and Jennifer L. Welch. Self-stabilizing clock synchronization in the presence of byzantine faults. *J. ACM*, 51(5):780–799, 2004.

[7] Swan Dubois, Toshimitsu Masuzawa, and Sébastien Tixeuil. The impact of topology on byzantine containment in stabilization. In *Proceedings of DISC 2010*, Lecture Notes in Computer Science, Boston, Massachusetts, USA, September 2010. Springer Berlin / Heidelberg.

[8] Swan Dubois, Toshimitsu Masuzawa, and Sébastien Tixeuil. On byzantine containment properties of the min+1 protocol. In *Proceedings of SSS 2010*, Lecture Notes in Computer Science, New York, NY, USA, September 2010. Springer Berlin / Heidelberg.

[9] Sukumar Ghosh, Arobinda Gupta, Mehmet Hakan, Karaata Sriram, and V. Pemmaraju. Self-stabilizing dynamic programming algorithms on trees. In *in Proceedings of the Second Workshop on Self-Stabilizing Systems*, pages 11–1, 1995.

[10] Wayne Goddard, Stephen T. Hedetniemi, David Pokrass Jacobs, and Pradip K. Srimani. Self-stabilizing protocols for maximal matching and maximal independent sets for ad hoc networks. In *IPDPS*, page 162, 2003.

[11] Wayne Goddard, Stephen T. Hedetniemi, and Zhengnan Shi. An anonymous self-stabilizing algorithm for 1-maximal matching in trees. In *PDPTA*, pages 797–803, 2006.

[12] Su-Chu Hsu and Shing-Tsaan Huang. A self-stabilizing algorithm for maximal matching. *Inf. Process. Lett.*, 43(2):77–81, 1992.

[13] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[14] Fredrik Manne, Morten Mjelde, Laurence Pilard, and Sébastien Tixeuil. A new self-stabilizing maximal matching algorithm. *Theoretical Computer Science (TCS)*, 410(14):1336–1345, March 2009.

[15] Fredrik Manne, Morten Mjelde, Laurence Pilard, and Sébastien Tixeuil. A self-stabilizing 2/3-approximation algorithm for the maximum matching problem. *Theoretical Computer Science (TCS)*, 412(40):5515–5526, September 2011.

[16] Toshimitsu Masuzawa and Sébastien Tixeuil. Stabilizing link-coloration of arbitrary networks with unbounded byzantine faults. *International Journal of Principles and Applications of Information Science and Technology (PAIST)*, 1(1):1–13, December 2007.

[17] Mikhail Nesterenko and Anish Arora. Tolerance to unbounded byzantine faults. In *21st Symposium on Reliable Distributed Systems (SRDS 2002)*, pages 22–29. IEEE Computer Society, 2002.

[18] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

[19] Gerard Tel. Maximal matching stabilizes in quadratic time. *Inf. Process. Lett.*, 49(6):271–272, 1994.

[20] Sébastien Tixeuil. *Algorithms and Theory of Computation Handbook, Second Edition*, chapter Self-stabilizing Algorithms, pages 26.1–26.45. Chapman & Hall/CRC Applied Algorithms and Data Structures. CRC Press, Taylor & Francis Group, November 2009.