Multiagent Plan Repair by Combined Prefix and Suffix Reuse

Antonín Komenda (komenda@agents.fel.cvut.cz)
Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic

Peter Novák (P.Novak@tudelft.nl)
Department of Software and Computer Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, The Netherlands

Michal Pěchouček (pechoucek@agents.fel.cvut.cz)
Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic

Abstract

Deterministic domain-independent multiagent planning is an approach to coordination of cooperative agents with joint goals. Provided that the agents act in an uncertain and dynamic environment, such plans can fail. The straightforward approach to recover from such situations is to compute a new plan from scratch, that is to replan. Even though, in a worst case, plan repair or plan re-use does not yield an advantage over replanning from scratch, there is a sound evidence from practical use that approaches trying to repair the failed original plan can outperform replanning in selected problems. One of the possible plan repairing techniques is based on preservation of fragments of the older plans.

This work theoretically analyses complexity of plan repairing approaches based on preservation of fragments of the original plan and experimentally studies three practical aspects affecting its efficiency in various multiagent settings. We focus both on the computational, as well as the communication efficiency of plan repair in comparison to replanning from scratch and we report on the influence of the following properties on the efficiency of plan repair: (1) the number of involved agents in the plan repairing process, (2) inter-dependencies among the repaired actions, and finally (3) particular modes of re-use of the older plans.

Keywords: multiagent systems, automated planning, plan repair

1 Introduction

Consider a team of heterogeneous robots working together so as to execute a mission in an environment. Since the robots feature heterogeneous capabilities, it might well be that none of them is able to complete the mission on its own, however by a careful coordination and teamwork, they should be able to reach the joint objective. The team of physical robots is embodied in a dynamic environment in which various events and plan execution interruptions occur and most importantly, in which actions of the agents can fail. To execute their mission, the robots represented by deliberative agents must be able to cope with such a dynamics on both, the individual, as well as the coordination level. Here we focus on the problem of multiagent plan repair which tackles such issue.

Recently, an approach of *multiagent (MA) plan repair* (MA-REPAIR) was proposed in [8], based on multiagent planning (MA-STRIPS) as introduced in [2] and the classical MODDELINS principle from [12]. MA-STRIPS is an approach to planning for teamwork and coordination extending the classical STRIPS-based planning techniques. According to the MA-REPAIR approach, the multiagent team computes a team plan using a fully decentralized MA-STRIPS planning algorithm, and subsequently executes the plan, while at the same time monitoring of possible failures of plan execution. Upon an occurrence of such a failure, the team stops execution and invokes a plan repair algorithm and fixes the failed joint plan in order to reach a joint goal state from the state in which the failure occurred.

It can be argued that plan re-use based on MODDELINS does not yield much advantage with respect to the computational complexity in the worst case [12], since attempts to fix a failed plan sometimes lead to replanning from scratch anyway. In multiagent and multi-robot settings, where communication is unreliable and costly, however, it is often the communication which is of higher priority than the computational complexity.

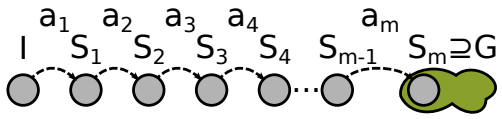
In [10], the authors have proposed prefix and suffix-based approaches to multiagent plan repair. These repairing approaches save communication in contrast to replanning from scratch in tightly coupled problems with action failures, however a research question which plan repairing techniques are more appropriate for which planning domains and problems remained unanswered. In this work, we extend our recent experimental study [9], where we have generalized the prefix and suffix-based approaches and present a coherent analysis of computational complexity and practical properties of how particular multiagent plan repair techniques and particular parameterizations perform in different planning domains.

2 Multiagent Planning & Repair

We use MA-STRIPS [2] as a model for the multiagent planning problems.

Definition 1. Let a quadruple $\Pi = \langle P, \mathcal{A}, I, G \rangle$ be a *multiagent planning problem* over

- a finite set of propositions P denoting facts about the environment the agents operate in,



$$\mathcal{P} = \begin{pmatrix} a_{11} & a_{21} & a_{31} & a_{41} & \cdots & a_{m1} \\ a_{12} & a_{22} & a_{32} & a_{42} & \cdots & a_{m2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & a_{3n} & a_{4n} & \cdots & a_{mn} \end{pmatrix}$$

Figure 1: A multiagent plan in matrix form with a visual representation of the intermediate states $S_1, \ldots, S_{m-1}$ and joint actions $\mathbf{a}_i = (a_{i1}, \ldots, a_{in})$. The gray nodes represent planned states of evolution of the environment, I is the initial state and G is the set of goal conditions defining a set of possible goal states. The arcs represent the planned joint actions.

- a set of n agents $\mathcal{A} = \{A_1, \ldots, A_n\}$, each characterized by a finite set of actions with STRIPS syntax and semantics the agent can perform, formally $A_i = \big\{\langle \mathsf{pre}(a), \mathsf{add}(a), \mathsf{del}(a)\rangle \mid \mathsf{pre}(a), \mathsf{add}(a), \mathsf{del}(a) \subseteq P\big\}$, where $\mathsf{pre}(a)$,$\mathsf{add}(a)$, and $\mathsf{del}(a)$ represent sets of preconditions, add effects, and delete effects respectively; the transition function describing change of the environment in state S after execution of action a into new state S' is defined as $S' = (S \cup \mathsf{add}(a)) \setminus \mathsf{del}(a)$ provided that $\mathsf{pre}(a) \subseteq S$, i.e., the action a is *applicable* in the state S,
- an initial state $I \subseteq P$ the environment begin in, and
- a goal state conditions $G \subseteq P$ characterizing agents' joint objective(s) s.t. a state $S \subseteq P$ is a goal state iff $G \subseteq S$.

Additionally, we define a set of propositions of an action a as $\mathsf{prop}(a) = \mathsf{pre}(a) \cup \mathsf{add}(a) \cup \mathsf{del}(a)$ and a set of propositions an agent A_i affects or is affected by as $P_i = \bigcup_{a \in A_i} \mathsf{prop}(a)$. Each action set also contains a empty action $\epsilon = \langle \emptyset, \emptyset, \emptyset \rangle$. According to MA-STRIPS, a distinguished subsets of agents' *public* actions known to all other agents is defined as $A_i^{\mathrm{pub}} = \{a | a \in A_i \,\mathrm{s.t.}\, \exists j \neq i : \mathsf{prop}(a) \cap P_j \neq \emptyset\}$ and the complement denoted as *private* actions is defined $A_i^{\mathrm{priv}} = A_i \setminus A_i^{\mathrm{pub}}$. A *joint action* of all agents is defined as a n-tuple $\mathbf{a} = (a_1, \ldots, a_n)$ where for each a_i holds $a_i \in A_i$. Execution of a joint action $\mathbf{a}$ is defined as $S' = (S \cup \bigcup_{a \in \mathbf{a}} \mathsf{add}(a)) \setminus \bigcup_{a \in \mathbf{a}} \mathsf{del}(a)$ provided that $\bigcup_{a \in \mathbf{a}} \mathsf{pre}(a) \subseteq S$ and $\bigcup_{a \in \mathbf{a}} \mathsf{add}(a) \cap \bigcup_{a \in \mathbf{a}} \mathsf{del}(a) = \emptyset$.

Definition 2. A sequence of joint actions $\pi = (\mathbf{a}_1, \ldots, \mathbf{a}_m)$ is a *multiagent plan* solving a multiagent planning problem $\Pi = \langle P, \mathcal{A}, S_0, G \rangle$ iff $\bigcup_{a \in \mathbf{a}_i} \mathsf{pre}(a) \subseteq S_{i-1}$ and $\bigcup_{a \in \mathbf{a}_i} \mathsf{add}(a) \cap \bigcup_{a \in \mathbf{a}_i} \mathsf{del}(a) = \emptyset$ where for the *intermediate states* hold $S_i = (S_{i-1} \cup \bigcup_{a \in \mathbf{a}_i} \mathsf{add}(a)) \setminus \bigcup_{a \in \mathbf{a}_i} \mathsf{del}(a)$ for $i \in 1, \ldots, m$ and $G \subseteq S_m$.

We will denote the length of a sequence of joint actions $\pi = (\mathbf{a}_1, \ldots, \mathbf{a}_m)$ as $|\pi| = m$. A concatenation of two multiagent plans will be denoted as $(\mathbf{a}_1^1, \ldots, \mathbf{a}_k^1) \cdot (\mathbf{a}_1^2, \ldots, \mathbf{a}_l^2) = (\mathbf{a}_1^1, \ldots, \mathbf{a}_m^1, \mathbf{a}_1^2, \ldots, \mathbf{a}_l^2)$. To index a k-th joint action in π, we will write $\pi[k]$ and $\pi[k, \ldots, l]$ where $1 \leq k \leq l \leq m$ will denote a fragment of the sequence $(\mathbf{a}_k, \ldots, \mathbf{a}_l)$. A public projection of plan π will be denoted as π^{pub} and replaces all

actions which are not in any $a \in A_i^{\text{pub}}$ by the empty action ϵ. A visual representation of a multiagent plan is depicted in Figure 1.

With formally defined multiagent planning problems and multiagent plans solving them, we can formally present a problem of multiagent plan repair MA-REPAIR as defined in [10]:

Definition 3. Let a quadruple $\Sigma = \langle \Pi, \pi, F, k \rangle$ be a *multiagent plan repair* problem over

- a multiagent planning problem $\Pi = \langle P, \mathcal{A}, I, G \rangle$,
- an original multiagent plan π solving the problem Π which execution failed s.t. after execution of some $\mathbf{a}_k$ the resulting state differs from the intermediate state S_k,
- a state $F \subseteq P$ which the system happens to be in, unexpectedly after the plan execution failure, and
- the step $k \in 1, \ldots, |\pi|$, after which the failure occurred.

A solution to a multiagent plan repair problem $\Sigma = \langle \Pi, \pi, F, k \rangle$ is a multiagent plan π' solving a modified planning problem $\Pi' = \langle P, \mathcal{A}, F, G \rangle$.

Two auxiliary definitions are needed for formal description of the proposed plan repair algorithm.

Definition 4. Let forward *proposition propagation* operator $\oplus$ be a mapping $\oplus : 2^P \times (\mathbf{A}_1 \times \mathbf{A}_2 \times \cdots \times \mathbf{A}_m) \to 2^P$, where each $\mathbf{A} = \times_{i=1}^{n} A_i$ and $m \geq 1$, defined as

$$S \oplus \pi \mapsto \begin{cases} (S \cup \bigcup_{a \in \pi[1] \text{ s.t. } \mathsf{pre}(a) \subseteq S} \mathsf{add}(a)) \setminus \bigcup_{a \in \pi[1] \text{ s.t. } \mathsf{pre}(a) \subseteq S} \mathsf{del}(a) & \text{for } |\pi| = 1, \\ (S \oplus (\pi[1])) \oplus \pi[2, \ldots, |\pi|] & \text{otherwise.} \end{cases} \tag{1}$$

Similarly to the transformation operator $\oplus$, a reverse-transformation operator is defined as follows:

Definition 5. Let *proposition back-propagation* operator $\ominus$ be a mapping $\ominus : 2^P \times (\mathbf{A}_1 \times \mathbf{A}_2 \times \cdots \times \mathbf{A}_m) \to 2^P$, where each $\mathbf{A} = \times_{i=1}^{n} A_i$ and $m \geq 1$, defined as

$$S \ominus \pi \mapsto \begin{cases} (S \cup \bigcup_{a \in \pi[1]} \mathsf{del}(a)) \setminus \bigcup_{a \in \pi[1]} \mathsf{add}(a) & \text{for } |\pi| = 1, \\ (S \ominus (\pi[|\pi|])) \ominus \pi[1, \ldots, |\pi| - 1] & \text{otherwise.} \end{cases} \tag{2}$$

For further use, we define a metrics on actions in a multiagent plan describing importance of the action with respect to the number of actions, which would be no longer applicable if the action is not present in the plan, formally:

Definition 6. Let $A_k^{\text{dep}} \subseteq \mathbf{a}_k$ in a multiagent plan π at step k. Let a set of actions A_{k+1}^{dep} contain all actions of $\mathbf{a}_{k+1}$ which are dependent on actions of A_k^{dep} based on the *effect–precondition relation*, formally $A_{k+1}^{\text{dep}} = \{a | a \in \mathbf{a}_{k+1}\ s.t.\ \bigcup_{a' \in A_k^{\text{dep}}} \mathsf{eff}(a') \cap \mathsf{pre}(a) \neq \emptyset\}$.

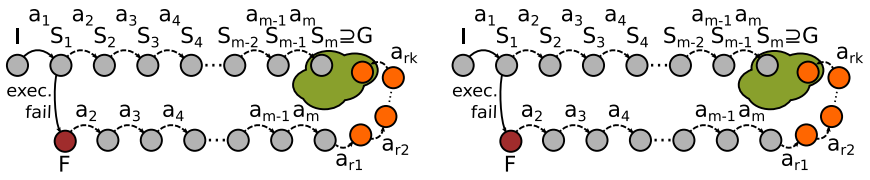


Figure 2: A visual representation of back-on-track (left) and lazy repair (right). The nodes and arcs has the same meaning as in Figure 1. The dashed arcs represent planned but not yet executed actions, the solid ones were already executed. Only the joint action $\mathbf{a}_1$ was executed without a failure. The execution of $\mathbf{a}_2$ failed and the environment ended in the state F. The orange nodes represent planned states of repair plan $(\mathbf{a}_{r1}, \ldots, \mathbf{a}_{rk})$. The pale orange states with their respective actions represent various possibilities of back-on-track repair plans. The cyan nodes represent a solution by replanning. In lazy repair, the application of $\mathbf{a}_2$ and further actions ignores preconditions, which might not be longer satisfied in F.

The *dependency* metrics dep of an action $a \in \mathbf{a}_k \in \pi$ is then defined as

$$\mathrm{dep}_\pi(a) \equiv |\bigcup_{i=k}^{|\pi|} A_i^{\mathrm{dep}} \; s.t. \; A_k^{\mathrm{dep}} = \{a\}|. \tag{3}$$

Besides straightforward multiagent replanning from scratch, which invokes a multiagent planner at the point of a failure and then executes the computed plan right away, two main approaches to multiagent plan repair were presented in [10]: *back-on-track* (BoT) and *lazy* repair (LR).

Both algorithms first formulate a modified multiagent planning problem and rely on the underlying multiagent planner to compute a plan fragment used for re-composition into a solution plan repairing the original failed one. We will refer to the underlying planner as an *inner planner* as it will be used as a component of the plan repair algorithm. The back-on-track strategy (see Figure 2left) tries to fix the prefix of the failed plan by computing a plan from the state in which the system happens to be right after the detection of a plan execution failure to some state along an ideal failure-free execution of the original plan. The resulting multiagent plan re-uses some *suffix* of the original plan, if possible, and extends the plan at its beginning. The idea underlying the lazy repair is complementary (see Figure 2right). Lazy repair takes the remainder of the original plan, re-uses all its actions which still can be executed according to their preconditions regardless of the outcome and completes the plan to some goal state of the planning problem. This way, the resulting plan is composed of re-used *prefix* parts of the original plan with an appended suffix of some new repaired plan. Experiments in [10] showed that these approaches lead to significant savings of communication, as well as computational resources in comparison to replanning from scratch on used planning domains and problems.

Based on the experimental analysis of the two plan repair algorithms and the formal definitions, we can state the core hypotheses of our work here.

Following the theoretical results of Brafman and Domshlak's complexity analysis of MA-STRIPS planning in [2], our motivation is not to substantially increase the computational complexity of the repair algorithms in respect to the inner planner. Although the communication complexity of the MA-STRIPS planning was not theoretically studied in the literature yet, we cover it in the hypothesis as well, as we hypothesize that the communication complexity is a function of the computational one.

Hypothesis 1. *Suffix, prefix or combined multiagent plan repair does not introduce any additional exponential dependency on number of involved agents in respect to the inner planner both in computational and communication complexity.*

Albeit the theoretical results showing that MA-STRIPS planning is not exponentially dependent on the number of involved agent, we hypothesize that practically the overhead of planning or plan repair with higher number of agents grows, especially if the inner planner assumes only public goal condition propositions, which is a common assumption in literature (e.g., in [13]).

Hypothesis 2. *Repairing algorithms minimizing the number of agents involved in the plan repairing process tend to generate lower computational and communication overheads than other strategies.*

Immediate repairing of actions, which substantial number of other actions dependent on (see Definition 6), is intuitively more efficient than repair of such actions later with possibly smaller reusable parts of the original plan and higher number of actions to repair in general. Next hypothesis describes this intuition.

Hypothesis 3. *Repairing algorithms reusing the original plan as a suffix generate lower computational and communication overheads than the repairing algorithms reusing the original plan as a prefix in domains with actions with high values of the dependency metrics.*

If we parameterize the lengths of reused prefix u and suffix v of the plan repairing process, an interesting question is, how do different combinations of these parameters influence the efficiency of the plan repairing process. Let m be the length of the reusable part of the original plan π, then $m = |\pi| - k$, where k is the step after the failed action. Obviously, for $u + v < m$, there will be a gap, which has to be filled by a result of the inner planner, in other words the original plan was *underused*. Reversely, for $u + v > m$, there will be an overlap, which has to be reverted, i.e., the original plan was *overused*. Intuitively, these cases are in a sense pathological. The last hypothesis states that repair strategies not underusing nor overusing the original plan should be the most efficient:

Hypothesis 4. *Repairing algorithms overusing or underusing the original plan tend to generate higher computational overheads than other algorithms.*

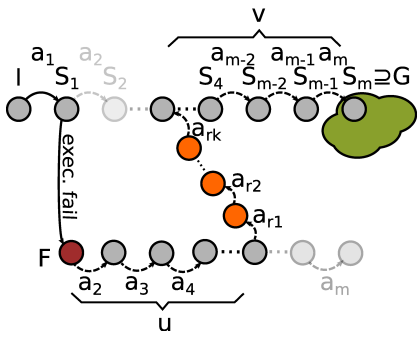


Figure 3: A visual representation of generalized repair. The nodes and arcs have the same meaning as in Figure 2. The repair plan $(\mathbf{a}_{r1}, ..., \mathbf{a}_{rk})$ is used to connect the prefix and suffix of the original plan. The parameters u and v prescribe how many actions are reused as the prefix and the suffix respectively.

3 Generalized Repair

As outlined in the previous section, the algorithm used in the further analyses is a combination of the lazy and back-on-track approaches, presented in [10], which are orthogonal to each other in how they reuse the original plans. These two approaches can be combined into one algorithm using the original plan both as a prefix and a suffix together. Such an approach *generalizes* the first two approaches and combines the original plan by both fashions as shown in Figure 3.

Definition 7. (generalized repair) Let $\Sigma = \langle \Pi, \pi, F, k \rangle$ be a multiagent plan repair problem and let $\Pi' = \langle P, \mathcal{A}, F, G \rangle$ be the corresponding modified multiagent replanning problem.

A multiagent plan π' solving Π' is a *generalized repair* of π for vectors of indexes U and V iff there is a decomposition of π', such that $\pi' = \bar{\pi}_F[k, \ldots, k+u] \cdot \pi_{\text{fix}} \cdot \pi[|\pi|-v, \ldots, |\pi|]$, where $\bar{\pi}_F[k, \ldots, k+u]$ is a fragment of plan π omitting inapplicable actions beginning with the state F for some $u \in U$, π_{fix} is a new plan connecting the reused fragments, and $\pi[|\pi| - v, \ldots, |\pi|]$ is a fragment of the original plan for some $v \in V$. The vectors contain only valid indexes to the reused part of the multiagent plan π, that is $\forall u \in U : 0 \leq u \leq |\pi| - k$ and $\forall v \in V : 0 \leq v \leq |\pi| - k$, to meet the requirements of the decomposition parts $\bar{\pi}_F[k, \ldots, k+u]$ and $\pi[|\pi| - v, \ldots, |\pi|]$. It holds that $|U| = |V|$ and for $\forall i$ s.t. $1 \leq i \leq |\pi|$ there is no $j : 1 \leq j \leq |\pi|, j \neq i$ s.t. $(u_i, v_i) = (u_j, v_j)$. Also $0 \in U, 0 \in V$.

To illustrate the generalized notion of this repair, it can be shown that for $U = (|\pi| - k), V = (0)$ the approach becomes the lazy repair and for $U = (0), V = (|\pi| - k, \ldots, 0)$ the back-on-track approach. In the first case, the original plan is reused as a plan fragment ignoring preconditions of length $|\pi| - k$ equally to the definition of lazy repair in [10]. In the other case, the definition of the index vector V implies trying to reuse a plan fragment starting with length $|\pi| - k$ and ending with length 0, equally to the back-on-track repair definition in [10]. Finally, $U = (0), V = (0)$ describes

replanning.

It could be argued that generalization reusing the original plan only as prefix and suffix parts is in fact not the only possible repair scheme, *e.g.*, by means of the MODDELINS scheme presented in [12]. The MODDELINS reuse scheme describing the presented generalized repair approach is

$$(\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_u) \cdot \pi_{\mathrm{fix}} \cdot (\mathbf{a}_{|\pi|-v}, \mathbf{a}_{|\pi|-(v-1)}, \ldots, \mathbf{a}_{|\pi|}), \tag{4}$$

however a scheme

$$(\mathbf{a}_1, \ldots, \mathbf{a}_u), \pi^1_{\mathrm{fix}}, (\mathbf{a}_{u+1}, \ldots, \mathbf{a}_{u+v}), \pi^2_{\mathrm{fix}}, (\mathbf{a}_{|\pi|-w}, \ldots, \mathbf{a}_{|\pi|}) \tag{5}$$

would be also possible, but generalized repair cannot directly represent it. Since the motivation of the generalized repair is not to be general in the sense of reuse pattern of the original plan, but be general from perspective of reuse of the original plan as both prefix and suffix plan fragments. For such case, the generalized repair scheme is complete, since breaking the $\bar{\pi}_F[k, \ldots, k+u]$ or the $\pi[|\pi|-v, \ldots, |\pi|]$ fragments into smaller parts would require additional π_{fix} plans, which could be concatenated into one central fixing fragment as defined in Definition 7.

The algorithm for the generalized repair is outlined in Algorithm 1. In the case, a failure is detected by the agent team, the current state after the failure is retrieved and the plan repair algorithm for the plan repair problem $\Sigma = \langle \Pi, \pi, F, k \rangle$ is invoked. In each plan repair attempt a modified multiagent planning problem is formulated according to the current values of u and v prescribing the length of the reused prefix and suffix of the original plan. These parameters are took from two vectors U and V parameterizing the repair.

If a repair plan is found, the repair process finishes, otherwise another attempt with a different combination of u and v is made (selection of u has priority over v in the combination of indexes). The resulting repairing plan consists of three components: the preserved prefix of the original plan π_{pre}, a newly computed infix π_{fix} and suffix part π_{suf}, again preserving a part of the original plan π.

The preserved prefix part $\bar{\pi}_F$ of the original plan corresponds to a plan fragment of π ignoring the preconditions such that only actions applicable in sequence beginning from state F are used. The actions with unmet preconditions are simply omitted. Additionally, the prefix π_{pre} is based only on a part of the original plan effectively reusing u actions beginning after the k-th action of the original plan π. The suffix part π_{suf} is obtained as the last v actions of the original plan π.

Finally, the infix part of the plan is computed by invocation of the inner multiagent planner MA-Plan[1]. The initial state of the modified planning problem is the state in which a failure-free execution of the repair prefix π_{pre} would result in starting from the state F, that is propagation $F \oplus \pi_{\mathrm{pre}}$. The set of goal states $G \ominus \pi_{\mathrm{suf}}$ corresponds to a back-propagation of effects of the preserved suffix component π_{suf} from the set of original goals G.

If the multiagent planner finds a plan for the modified planning problem, the repair plan takes the form $\pi_{\mathrm{pre}} \cdot \pi_{\mathrm{fix}} \cdot \pi_{\mathrm{suf}}$ and gets executed from the failure point on. In

[1] In the experiments, we used the Planning First implementation of a MA-STRIPS planner from [13].

Algorithm 1 Generalized-Repair(Σ, U, V,MA-Plan)

Input: A multiagent plan repair problem$\Sigma = \langle \Pi, \pi, F, k \rangle$.

Input: Parameters U and V prescribing the lengths for reusing of the original plan as prefix and suffix respectively.

```
1: u, v =initial pair of u ∈ U and v ∈ V
2: repeat
3:    π_pre = π̄_F[k, ..., k + u]
4:    π_suf = π[|π| − v, ..., |π|]
5:    π_fix = MA-Plan(⟨P, A, F ⊕ π_pre, G ⊖ π_suf⟩)
6:    if π_fix ≠ ∅ then
7:       π = π_pre · π_fix · π_suf
8:       break
9:    end if
10: until tested all pairs ofu ∈ U andv ∈ V
11: ifπ = ∅ then return fail
```

the case no repair plan can be found, the algorithm attempts the repair for a different combination of u and v until either a repair plan is found, or it turns out that no repair for the failure exists.

The description of the algorithm will be concluded with proofs of soundness and completeness. Since the generalized repair algorithm uses inner invocation of the inner multiagent planner similarly to back-on-track and lazy repairs [10], its correctness relies on the correctness of the inner planner.

Lemma 8. *(soundness). Let $\Pi = \langle P, \mathcal{A}, I, G \rangle$, be a multiagent planning problem with agents situated in a dynamic environment in which the environment can interfere with the plan execution and let π be a solution to Π. Let also F be a state resulting from an interference of the environment, a plan failure, at a step k of execution of the plan π. $\Sigma = \langle \Pi, \pi, F, k \rangle$ denotes the corresponding multiagent plan repair problem.*

Provided that the execution of Generalized-Repair(Σ,U,V,*MA-Plan*) *does not fail, but finishes with a resulting plan π' and MA-Plan is a sound* MA-STRIPS *planner, a failure-free execution of π' leads to some goal state of the original multiagent planning problem Π.*

Proof. Regardless what particular state $F \oplus \pi_{\text{pre}}$ the failure-free execution of the applicable actions from the fragment $\bar{\pi}_F$ ends up in, the solution plan π_{fix}, if exists, to the problem $\langle P, \mathcal{A}, F \oplus \pi_{\text{pre}}, G \ominus \pi_{\text{suf}} \rangle$ will take the system from the state $F \oplus \pi_{\text{pre}}$ to a state $G \ominus \pi_{\text{suf}}$ corresponding to the original multiagent planning problem Π. The back-propagated propositions $G \ominus \pi_{\text{suf}}$ either represent a required (part of) state along the original execution trace of the original plan π and then the remainder π_{suf} leading to an original final state is reused, or a failure-free execution of π_{fix} leads directly to one of the final states defined by G without reusing a part of π as suffix $\pi_{\text{suf}} = \emptyset$, and therefore $G \ominus \pi_{\text{suf}} = G$. □

The initial part and the final part of the proof is based on the soundness proofs of lazy repair and back-on-track respectively in [10]. As mentioned before, in general-

ized repair, these two approaches merge, therefore the proofs are based on the same argumentation.

Lemma 9. *(completeness). Let* $\Pi = \langle P, \mathcal{A}, I, G \rangle$ *be a multiagent planning problem and let* π*,*F*,*k*, as well as* Σ *are as in the Lemma 8. Let* U *and* V *be integer vectors by Definition 7.*

If there exists a solution plan to the multiagent planning problem $\Pi' = \langle P, \mathcal{A}, F, G \rangle$ *and* MA-Plan *is a complete* MA-STRIPS *planner, then the execution of* Generalized-Repair(Σ,U,V,MA-Plan) *finishes and finds a repair plan of*π.

Proof. The algorithm tests all combinations of U and V values. It eventually tests the required combination $u = 0$ and $v = 0$. Based on the definition of the algorithm, in such case, $\pi = \pi_{\mathrm{pre}} \cdot \pi_{\mathrm{fix}} \cdot \pi_{\mathrm{suf}}$ degenerates to $\pi = \pi_{\mathrm{fix}}$ since $\bar{\pi}_F[k, \ldots, k+u] = \bar{\pi}_F[k, \ldots, k] = \emptyset$ and $\pi[|\pi| - v, \ldots, |\pi|] = \pi[|\pi|, \ldots, |\pi|] = \emptyset$. The π_{fix} is then a solution of Π' generated by MA-Plan. If such solution exists, it is found, since MA-Plan is assumed to be complete. □

The principle of the completeness proof follows the idea of the back-on-track proof and in addition it requires fewer assumptions than the lazy repair which is complete only for dead-end free planning problems.

4 Complexity Analysis

In this section, the presented plan repair algorithm will be theoretically studied from perspective of a classical complexity metrics and one additional metrics suitable for distributed algorithms. The classically studied metrics is time complexity. Additionally, in multiagent systems, one can use a metrics based on an asymptotic ratio of communication volume required for an algorithm to finish to size of the input, similarly as in the case of the time complexity.

4.1 Time Complexity of MA-Plan

The generalized plan repair algorithm presented in the previous section use a multiagent planner as a component, therefore its complexity is an key part of the further analysis. The time complexity of the multiagent planning based on solving of coordination Constraint Satisfaction Problem (CSP) and internal heuristic search (and therefore the MA-Plan implementation of the planner presented in [13] as Planning First) was studied in [2], therefore the analysis of the time complexity from [2] will be recalled in the following paragraphs.

Informally, the complexity is *"the number of times [needed] to verify that a certain choice of coordination-sequence length forms a basis for a solution*$\times$ *the complexity of the verification process"*. To formally describe the time complexity of the MA-Plan approach, firstly [2] define size of the CSP domains for each agents' CSP variable. Each value of each domain represents one possible coordination sequence for one agent. Such sequences consist of at most δ coordination points defined as pairs (a, t) with a public action a and $1 \leq t \leq n\delta$ for n agents.

The idea of the $n\delta$ limit for the virtual time points t can be demonstrated on an example with $n = 2$ agents A_i, A_j,$i \neq j$ and $\delta = 3$ with precisely three used coordination points for both agents:

$$\begin{array}{l} A_i: \\ A_j: \end{array} \left(\begin{array}{cccccc} a_1^i & * & a_2^i & * & a_3^i & * \\ * & a_1^j & * & a_2^j & * & a_3^j \end{array} \right). \tag{6}$$

The example shows the longest possible coordination pattern for that particular instance as both the agents use all coordination points possible and the actions depends on each other such that no prolonging of the pattern is possible. In that case, the first coordination point is $(a_1^i, 1)$ and the last one $(a_3^j, n\delta)$, where $n\delta = 6$.

The size of the CSP domain as defined in [2] is for an agent A_i

$$|D_i| = \sum_{d=1}^{\delta} \left(\begin{array}{c} n\delta \\ d \end{array} \right) \cdot |A_i^{\mathrm{pub}}|^d = O((n\delta|A_i^{\mathrm{pub}}|)^{\delta+1}). \tag{7}$$

The term $\left(\begin{array}{c} n\delta \\ d \end{array} \right)$ represents all possible combinations of d virtual time points for the public actions (*e.g.*, for $d = 2, n\delta = 6$ there are 15 of them $\{(1,2),(1,3),\ldots,(1,6),(2,3),(2,4),\ldots,(5,6)\}$) and the term $|A_i^{\mathrm{pub}}|^d$ represents all possible public action sequences of length d of agent α (*e.g.*, for $d = 2$ and $|A_i^{\mathrm{pub}}| = 2$ and the sequences are $\{a_1a_1, a_1a_2, a_2a_1, a_2a_2\}$), therefore for each d, the complete term in the sum counts the number of possible coordination sequences for d coordination points. Finally, the summed up result represent the number of all possible coordination sequences for one agent.

The domain size is then used in the final time complexity formula for the *internal planning constraints* (ipc) in the CSP in the following form

$$O(f(\mathcal{I}) \cdot n \cdot \max_{A_i \in \mathcal{A}} |D_i|) = O(f(\mathcal{I}) \cdot n(n\delta|A^{\mathrm{pub}}|)^{\delta+1}) = O_{ipc}, \tag{8}$$

where the term $f(\mathcal{I})$ represents maximal complexity of individual planning $\mathcal{I}$ with a function f describing the cost of switching from regular planning and $A^{\mathrm{pub}} = \bigcup_{i=1}^{n} A_i^{\mathrm{pub}}$.

The complexity induced by the *coordination constrains* (cc) is in [2] derived from time complexity of Adaptive-Tree-Consistency algorithm (ATC) for solving CSP problems. The complexity is based on a tree-width ω of the CSP constraint graph [5] which is

$$O(n \cdot \max_{A_i \in \mathcal{A}} |D_i|^{\omega+1}) = O(n(n\delta|A^{pub}|)^{\delta\omega+\epsilon}) = O_{cc}, \tag{9}$$

where $\epsilon = \delta + \omega + 1$ is dominated by $\delta\omega$. According to [2], the constraint graph is isomorphic to the moral graph of agent interaction graph, therefore ω can be treated as a tree-width of the agent interaction graph. An *agent interaction graph* describes dependencies of the agents on each other defined by public actions.

The final complexity is a sum of the complexities from Eq. 8 and Eq. 9 for the particular constraints

$$O_{ipc} + O_{cc} = O(f(\mathcal{I}) \cdot n(n\delta|A^{pub}|)^{\delta+1} + n(n\delta|A^{pub}|)^{\delta\omega+\epsilon}) = O_{MAP}. \tag{10}$$

The complexity has no direct exponential dependence on the number of agents n, has no direct exponential dependence on the length of the individual plans of the agents and has no direct exponential dependence on the size of the original planning problem $|\Pi|$. However, the complexity of the individual planning $f(\mathcal{I})$ is in general still exponential in the size of the individual planning problems.

4.2 Time Complexity of the Generalized Repair Algorithm

Let $\Pi = \langle P, \mathcal{A}, I, G\rangle$ be the original multiagent planning problem and $\Pi' = \langle P, \mathcal{A}, F, G\rangle$ be the related multiagent replanning problem. The time complexity of the planning problemΠ is complexity of the MA-Plan algorithm as showed in the previous section

$$O(f(\mathcal{I}) \cdot n(n\delta q)^{\delta+1} + n(n\delta q)^{\delta\omega+\epsilon}), \tag{11}$$

where for brevity $q = |A^{pub}|$. Straightforwardly, the replanning complexity is in general the same as of planning, since the only difference is another initial state F. That means n, q and ω are the same[2]. The length δ depends on the particular initial state, however there is no guarantee that δ for F will be generally higher or lower then δ for I. From [12], it is known that plan reuse cannot be generally less complex than replanning from scratch.

Lemma 10. *(time complexity). Let $\Sigma = \langle \Pi, \pi, F, k\rangle$ be a multiagent plan repair problem and let $\Pi = \langle P, \mathcal{A}, I, G\rangle$ be the related multiagent planning problem. Let U and V be the index vectors of generalized plan repair by Definition 7.*

The asymptotic time complexity of Generalized-Repair(Σ,U,V,*MA-Plan*)

$$O_{GEN} = O(f(\mathcal{I}) \cdot l^2 n(n\delta q)^{\delta+1} + l^2 n(n\delta q)^{\delta\omega+\epsilon} + 2l^3\mathcal{G}^2 + 2l), \tag{12}$$

where $\mathcal{G} = |\Pi|$, $l = |\pi|$, $q = |A^{pub}|$, and the rest of the symbols follow the definitions for Eq. 10 from [2].

Proof. The Generalized-Repair algorithm is parametrized by two index vectors U and V which are used in a repeated search for a solution of the multiagent plan repair problem. The time complexity is informally *how many times the algorithm needs generate and test a repair strategy $\times$ what is the complexity of the generate and test procedure.*

The generate and test procedure consists of two proposition propagation procedures (by Definitions 4 and 5) each in worst case using simulation of time complexity, which in the worst case require l testings of all actions' $|A|$ possible preconditions $|P|$, therefore $O(l|A||P|) = O(l\mathcal{G}^2)$, since the number of actions and propositions cannot be bigger than the size of the whole planning problem. Technically, the extraction of the prefix fragment π_{pre} is part of the proposition propagation process $F \oplus \pi_{\mathrm{pre}}$, therefore there is one $O(l\mathcal{G}^2)$ term. Another $O(l\mathcal{G}^2)$ term is needed for the proposition back-propagation $G \ominus \pi_{\mathrm{suf}}$ similarly related to extraction of the suffix π_{suf}. The π_{fix} fragments requires solving one multiagent planning problem, hence the time complexity of one generate and test procedure is

$$O_{GT} = O_{MAP} + O(2l\mathcal{G}^2) = O(f(\mathcal{I}) \cdot n(n\delta q)^{\delta+1} + n(n\delta q)^{\delta\omega+\epsilon} + 2l\mathcal{G}^2). \tag{13}$$

[2]The difference in sizes of different states cannot be larger then $|P|$, which bounds it by a constant.

The procedure with complexity O_{GT} is in the Generalized-Repair used maximally $|U| \cdot |V|$ times. If a solution is found, two additional concatenations are needed to finally build the solution, therefore

$$\begin{aligned} O(|U| \cdot |V|) \cdot O_{GT} + O(|\pi_{\mathrm{pre}}|) + O(|\pi_{\mathrm{pre}}|) &= \\ O(l^2 \cdot f(\mathcal{I}) \cdot n(n\delta q)^{\delta+1} + l^2 \cdot n(n\delta q)^{\delta\omega+\epsilon} + 2l^3\mathcal{G}^2 + |\pi_{\mathrm{pre}}| + |\pi_{\mathrm{suf}}|) &= \\ O(f(\mathcal{I}) \cdot l^2 n(n\delta q)^{\delta+1} + l^2 n(n\delta q)^{\delta\omega+\epsilon} + 2l^3\mathcal{G}^2 + 2l) &= O_{GEN} \end{aligned} \tag{14}$$

where $|U| \cdot |V| = O(|\pi|^2) = O(l^2)$, as the index vectors can parametrize at most all combinations of the indices to the original plan π by Definition 7. The lengths of the resulting plan segments $|\pi_{\mathrm{pre}}|$ and $|\pi_{\mathrm{suf}}|$ are bounded by the length of the plan $l = |\pi|$. □

The resulting time complexity of generalized repair algorithm does not comprise any extra exponential dependency on any of the additional terms, which are always polynomial provided that *l is polynomial w.r.t. $\mathcal{G}$*. Consistently with [12], the asymptotic worst-case complexity is also never reduced, which is anticipated result of the analysis. The idea of the presented plan repair technique in general is of lowering δ by simplifying the inner planning process with help of reuse of parts of the original plan. Since δ is in O_{MAP} in two exponent terms, such idea is *positively supported by the analysis* as well.

The results of the time complexity analysis of the repair algorithms therefore *prove* the time complexity part of the first stated Hypothesis 1 with an additional assumption on the polynomial length of the repaired plan. Additionally, they are not in conflict with the remaining three hypotheses. Hypothesis 2 targets taking only a subset of $\mathcal{A}$ which can in effect lower the tree-width ω if the remaining agents are less coupled. In Hypothesis 3, the length δ of the inner repair plan is targeted, as in the cases of problems with actions having long dependency trees, it is theorized that fixing the problem sooner will require smaller δ than solving it later possibly with longer reverting plan of a bigger δ. In the last Hypothesis 4, smaller δ should be achieved by possibly short repair plans, where no reverting is caused by overusing of the original plans $u \in U, v \in V, u + v > l - k$ and no unnecessarily long repair plans are needed provided that $u + v < l - k$.

4.3 Communication Complexity of MA-Plan

For the study of communication complexity of the presented multiagent plan repair algorithm, firstly, the communication complexity of the inner planning process has to be known. Since the work of Brafman and Domshlak [2] formally tackle only time complexity, we propose a analysis of communication in a similar manner based on an analysis of the Adaptive-Tree-Consistency (ATC) algorithm.

The communication complexity of ATC can be derived from space complexity which was studied in [5]. The analysis will use the Big-O notation similarly to the previous section. To distinguish time and communication complexity, the Big-O will use superscript c for communication complexity O^c.

Size of each message in ATC is $\max_{A_i \in \mathcal{A}} |D_i|^{\mathrm{sep}}$, where sep is size of a maximal separator size in tree decomposition of the CSP. The size of the separator is in bucket-trees [5]

the tree-width ω, therefore a worst case size of one message is $\max_{A_i \in \mathcal{A}} |D_i|^{\omega}$. Maximal number of messages communicated in a bucket-tree CSP solver is double the number of arcs (two messages for each arc in tree of n vertex graph), therefore $2(n-1)$, where n is the number of the buckets. The buckets represent the CSP variables (with constraints in them resembling the principle of tree-decomposition), therefore the number of the buckets is the number of agents in the coordination constraint (cc). Since the internal planning constraints (ipc) are represented as unary constraints, they do not require any communication. All together, it gives the communication complexity of the MA-STRIPS planning as

$$2(n-1) \cdot \max_{A_i \in \mathcal{A}} |D_i|^{\omega} = O^c(\varepsilon n (n\delta q)^{\delta\omega+\epsilon} + \epsilon') = O^c(n(n\delta q)^{\delta\omega+\epsilon}) = O^c_{MAP}, \quad (15)$$

where ϵ is dominated by $\delta\omega$ in the exponent, $\varepsilon = 2$ is a polynomial coefficient and ϵ' is dominated by the first polynomial term. The communication complexity of planning using CSP for coordination is therefore not exponentially dependent on the number of agents n, it is not dependent on the complexity of the individual planning $\mathcal{I}$ and it has no direct exponential dependence on the size of the original planning problem $|\Pi|$, similarly to the time complexity. The communication complexity is bounded by one exponential term in the number of the coordination points and tree-width of the agent interaction graph

$$\exp(\delta\omega). \quad (16)$$

4.4 Communication Complexity of the Generalized Repair Algorithm

The communication complexity of generalized repair will be derived by an equal process as in the case of the time complexity. It builds on the derived complexity of the inner planning of MA-Plan which is in the case of communication $O^c(n(n\delta q)^{\delta\omega+\epsilon})$ as showed in Eq. 15.

Equally to the time complexity, replanning is in general the same as planning from the perspective of communication complexity. The only difference is in the initial state. That means n, q and ω are the same and δ depends on the particular initial state without any general guarantees on its change during replanning.

Lemma 11. *(communication complexity) Let $\Sigma = \langle \Pi, \pi, F, k \rangle$ be a multiagent plan repair problem and let $\Pi = \langle P, \mathcal{A}, I, G \rangle$ be the related multiagent planning problem. Let U and V be the index vectors of generalized plan repair by Definition 7.*

The asymptotic communication complexity of Generalized-Repair(Σ,U,V,*MA-Plan*)

$$O^c_{GEN} = O^c(l^2 n (n\delta q)^{\delta\omega+\epsilon} + 2nl^3\mathcal{G} + n^2 + n), \quad (17)$$

where $\mathcal{G} = |\Pi|$, $l = |\pi|$, $q = |A^{pub}|$, and the rest of the symbols follow the definitions for Eq. 10 from [2].

Proof. In the Generalized-Repair, the process is separable to repeated sub-processes of generating and testing of a repair strategy and it needs an additional synchronization

broadcast as the agents has to be aware of new plan repair process in case of failure in a private fact. Such broadcast is an additive factor $O^c_{\mathrm{sync}} = O(n)$ as the messages are sent to all agents.. The communication complexity will be firstly derived for one such sub-process. The two proposition propagation procedures $F \oplus \pi_{\mathrm{pre}}$ and $G \ominus \pi_{\mathrm{suf}}$, each in the worst case use the same principle as a distributed simulation of execution of a multiagent plan segment $O(nl|P|) = O(nl\mathcal{G})$ and are used with one inner planning, therefore

$$O^c_{GT} = O^c(n(n\delta q)^{\delta\omega+\epsilon} + 2nl\mathcal{G}). \tag{18}$$

Equally to the time complexity, the sub-process with the complexity O^c_{GT} can be used in worst case $|U| \cdot |V|$ times. If a sound solution is found the agents has to inform each other, therefore there is beside the initial synchronization a termination synchronization of $O^c(n^2)$, although the local plans has not to be communicated as they are later executed by the particular agents. The communication complexity of the Generalized-Repair is

$$\begin{aligned} O^c_{\mathrm{sync}} + O^c(|U| \cdot |V|) \cdot O^c_{GT} + O^c(n^2) &= \\ O^c(n + l^2 n(n\delta q)^{\delta\omega+\epsilon} + 2nl^3\mathcal{G} + n^2) &= \\ O^c(l^2 n(n\delta q)^{\delta\omega+\epsilon} + 2nl^3\mathcal{G} + n^2 + n) &= \quad O^c_{GEN}, \end{aligned} \tag{19}$$

where $|U| \cdot |V| = O(|\pi|^2) = O(l^2)$, as the index vectors can parametrize at most all combinations of indices to the original plan π by Definition 7. □

The analyzed communication complexity do not bring any new terms exponentially dependent on any of the parameters, which are always polynomial provided that *l is polynomial w.r.t. $\mathcal{G}$*. Therefore the communication complexity of the proposed plan repair algorithm remains exponential only in the factor of number of coordination points δ in the inner repair plan and tree-width ω of the agent interaction graph, i.e., $\exp(\delta\omega)$ as in Eq. 16. This result is anticipated as the communication complexity is usually proportional to the time complexity.

The resulting communication complexity of the generalized repair algorithm *proves* the communication and final part of Hypothesis 1 with an additional assumption on the polynomial length of the repaired plan. Additionally, the results support the experimental results from [10] and concur with the remaining hypotheses, similarly as in the case of the time complexity. The core hypothesis of [10] states that the communication overhead is lowered by plan repair producing more preserving repairs in comparison to replanning. Since the communication complexity of replanning is exponentially dependent on δ this hypothesis is supported by the analysis as far as at least one coordination point is spared, because decreasing the exponential factor by one $\exp((\delta - 1)\omega)$ dominates any additional polynomial factors added by the plan repair techniques. This is true only, if the problems are tightly coordinated $\omega \gg 0$. If it is the contrary, the exponential factor is negligible even if δ is not decreased by the preservation of the repair, formally $\exp(\delta\omega) \to 1$ iff $\delta \to 0$ or $\omega \to 0$.

The arguments on decreasing the amount of communication used for the remaining three hypothesis of this article copies those in the time complexity analysis. Hypothesis 2 targets taking only a subset of $\mathcal{A}$, which can in effect lower the tree-width ω if the

Algorithm 2 Plan execution and monitoring scheme.

Input: An initial multiagent planning problem $\Pi = \langle P, \mathcal{A}, I, G \rangle$, vectors U and V, and Planning First multiagent planner by [13] as MA-Plan.

```
1: π = MA-Plan(Π)
2: if MA-Plan failed then return fail
3: k = 1
4:
5: repeat
6:    agents perform π[k]
7:    if failure detected then
8:       retrieve the current state F from the environment
9:       π = Generalized-Repair(⟨Π, π, F, k⟩, F, G, MA-Plan)
10:      k = 1
11:   else
12:      k = k + 1
13:   end if
14: until Generalized-Repair failed or k > |π|
```

remaining agents are less coupled, and therefore lower the communication complexity. In Hypothesis 3, the length δ of the inner repair plan should be minimized if failures of actions with long dependency trees are fixed as soon as possible. In Hypothesis 4, smaller δ should be achieved by possibly short repair plans by appropriate reusing of the original plan.

5 Experimental Analysis

The further sections of the article focus on the remaining hypotheses and its experimental analysis.

The experiments were conducted in a synthetic setting, a simulated world with a group of agents using a plan execution, monitoring and repair loop (see Algorithm 2). The world was modeled as fully observable. All failures of plan execution were generated by the simulator according to a uniform distribution over time and parametrized by a probability p of failure occurrence in each step for each experiment. The failures were handled by the agents immediately upon detection.

A failure was simulated by not-execution of some of the agent actions from the actual plan step. The particular actions were chosen according to an uniform probability distribution over the individual actions within a joint action. As showed by [10], failure models with more radical impacts on the environment (e.g., state perturbations) decrease usability of the plan repairing approaches. Our motivation in this work is to study types of plan repairing strategies, therefore we stick only to action failures.

For the implementation of the experimental setup and the repairing algorithms, we used a centralized world simulator integrating the multiagent domain-independent planner Planning First [13] denoted as MA-Plan. Each agent run in its own thread

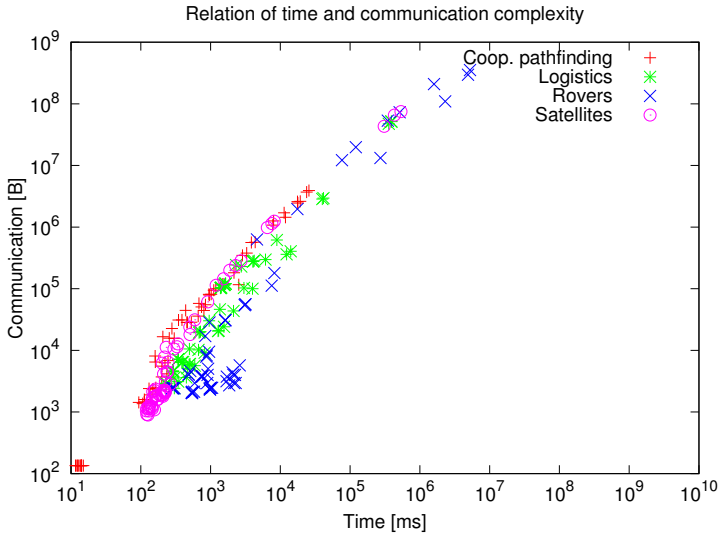


Figure 4: Relation between communicated bytes and computation time for solving the plan repairing problems.

and deliberated asynchronously. The experiments were executed on 8-core processor at 3.6GHz with Java Virtual Machine limited to 2.5GB of RAM.

For the experiments, we used four planning domains. Three of them originate in the standard single-agent IPC planning benchmarks. Similarly to the evaluation of the MA-Plan algorithm in [13], we chose domains, which are straightforwardly modifiable to a multiagent setting: LOGISTICS, ROVERS, and SATELLITES. Additionally, we have extended the set of benchmarks by COOPERATIVE PATHFINDING coordination domain on a grid [10].

The experimental measurements were based on two metrics focusing on the target efficiencies: cumulative time consumed by the particular plan repairing algorithms during a single run of the simulation, i.e., the overall time spent in the algorithm (incl. the underlying planning process) excluding the initial planning phase of the scheme (Algorithm 1). The second metric was communication complexity of the process, that is the volume of communicated information in bytes among the involved agents during the plan repairing processes. Those are mainly the messages generated by the DisCSP solver of the Planning First MA-Plan planner and an additional synchronization processes minimizing the number of agents involved in the plan repairing process.

To account for differences in essential computational and communication complexity of the domains, we conducted a relationship experiment between these two measures. Figure 4 depicts the results and demonstrates that there is no essential discrepancy between the computational and communication complexity of the plan repairing solutions. That means the following results are not biased by problems extremely hard in time and simple in communication and vice versa.

5.1 The Number of Repairing Agents

Regardless of the theoretical results presented in [2], showing that the computational complexity of CSP-based multiagent planning is not exponentially dependent on the number of the agents, in practical experiments, we faced a non-negligible dependence of this number and required communication and computational effort. The first set of experiments analyzes this relation by means of Hypothesis 2.

5.1.1 Used Plan Repair

To validate Hypothesis 2, we have prepared an extensive set of plan repairing strategies stemming from the generalized repair. They can be divided into three main groups: one *without agent* count minimization, and two *with agent minimization*. First of the minimization groups reuse the original plan purely as a suffix and the other one purely as a prefix.

The differences of the strategies within one of the groups of repair strategies lies in the preference between agent minimization, size of preservation of the original plan and bound on the maximal length of the newly generated repairing plan component π_{fix}. This approach restrain bias possibly caused by unbalanced influences of the agent minimization on various types of plan repair strategies.

The approach minimizing the number of involved agents was based on the notion of a set of *supporting agents*. The iterative process from Algorithm 3 was extended with an iteration starting only with a set of agents providing at least one action, which can contribute to the repairing plan by a required proposition(s), i.e., support part of $G \ominus \pi_{\mathrm{suf}}$. If such team of agents is not able to solve the plan repairing problem, the team is extended by additional agents supporting any of the current agents in the team by means of contributing to prepositions in their preconditions. If such additional agent does not exist and the team is still not containing all the agent from $\mathcal{A}$, a random agent is added into the team and the process continues.

5.1.2 Results and Discussion

The experiments were conducted in all presented planning domains and for all combinations of agent counts, i.e., two to four agents giving twelve domain and problem instances. Each of the group contained six variances of the repairing strategies giving 216 experiments in total. Each of the experiments was averaged over 5 measurements with different random seeds.

Figure 5 shows results of the first batch of experiments. The first group of repairing strategies not minimizing number of involved agents (red color) is in most measurements in both computational and communication metrics worse than the baseline replanning strategy. The suffix preserving algorithms minimizing numbers of agents (green color) is on the other hand nearly in all measurements better in both metrics than the baseline strategy with an exception in the simplest COOPERATIVE PATHFINDING problems. The group of plan repairing strategies minimizing the number of involved agents and preserving prefix part of the original plan (blue color) is on tie or better with the replanning in rather loosely coupled domains. The communication and computational overheads decrease with decreasing coupling of the domains. However in tighter

coupled domains, the strategies fall behind the replanning baseline. In LOGISTICS domain, only 33% of the strategies are better by communication overheads and only 18% by means of computational overheads. With increasing coupling the approach lose more. These results *support the second hypothesis*.

Additionally, the results revealed that the prefix preserving approaches, as not the best in all agent minimizing approaches, in most of the experiments has one of the best approaches outperforming the best suffix preserving approach. In LOGISTICS domain, the separation between the best prefix and best suffix preserving plan repairing strategy is about a half an order of magnitude in favor of the one prefix preserving approach. On the other hand, in COOPERATIVE PATHFINDING, suffix preserving approaches gain more than one order of magnitude.

5.2 Repairing of Actions with High Dependency Metrics

The intuition behind Hypothesis 3 can be rephrased as follows: If an action fails and it has potentially a lot of future dependencies, possibly of other agents or even in the goal, trying to fix it as soon as possible is rather better idea, than ignore it and try to repair it later. The experiments in this section were conducted to validate this concept.

5.2.1 Used Plan Repair

The most straightforward approach here is to compare the two plan repairing strategies re-using the whole original plan either as a prefix or as a suffix. These strategies are again parameterizations of the plan generalized repair such that there is no iteration over various u and v, but only two fixed values. The *pure prefix strategy* uses fixation $u = |\pi| - k, v = 0$ and the *pure suffix strategy* uses fixation $u = 0, v = |\pi| - k$.

In order to explain the result, we have to present more details on the LOGISTICS domain. In the LOGISTICS problem with three agents used in the experiments, the agents control two trucks T_1 and T_2 and one airplane A. There are two cities, each with one storage depot (d_1 andd_2) and one airport (a_1 anda_2). The trucks can move $\mathsf{m}(\mathsf{from}, \mathsf{to})$ only within their cities, i.e., between one depot and one airport. The airplane can fly $\mathsf{f}(\mathsf{from}, \mathsf{to})$ among all airports in the environment, but cannot land at the depots. All vehicles can load $\mathsf{l}(\mathsf{package}, \mathsf{location})$ and unload $\mathsf{u}(\mathsf{package}, \mathsf{location})$ a package at a location. Initially, there is one package p at one of the depots and the goal is to transport it to the other depot in the other city. The trucks start at the depots and the airplane starts at one of the airports. A typical multiagent plan solving this particular instance is depicted in the matrix form in Figure 6.

5.2.2 Results and Discussion

To validate Hypothesis 3, we run the pure prefix and pure suffix preserving repairing strategies in all the testing domains. We have measured ratio of successful repairs of these two repairing strategies against replanning by means of computation time. In Figure 7, we summarize the results of these experiments.

In the ROVERS and SATELLITE domains the plans solving the problem do not contain any significant actions by means of number of future dependencies to the overall

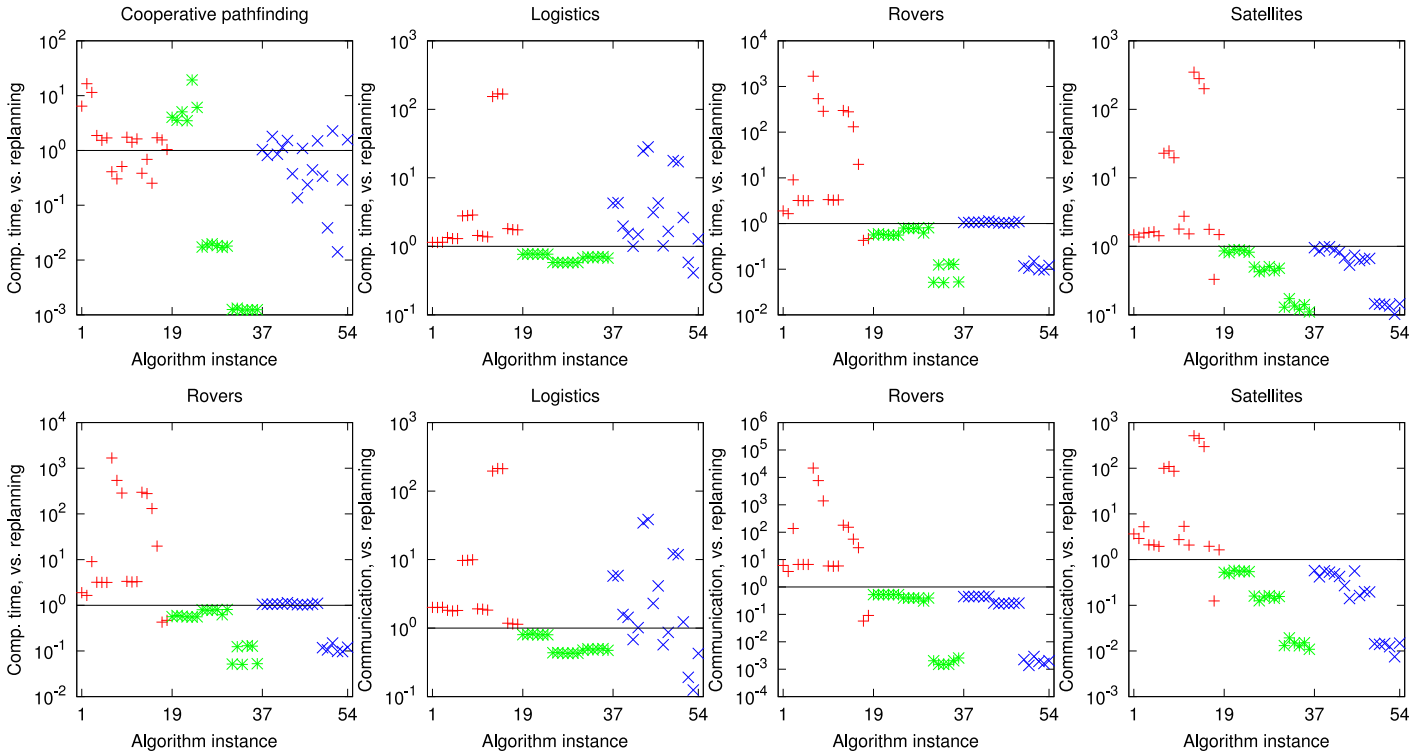


Figure 5: Comparison of various plan repairing strategies in proportion to replanning (black line at $y = 1$) with failure probability $p = 0.3$. Each point represent a mean of several runs of one of the particular repairing strategy. The red group contains plan repairing strategies using only the full set of agents involved in the original planning problem, the green group contains strategies using minimization of the number of agents involved and preserving suffix (back-on-track) of the original plan and the blue group contains strategies also minimizing number of agents and preserving prefix (lazy repair) of the original plan.

$$
\begin{array}{l}
\mathsf{A}: \\
\mathsf{T}_1: \\
\mathsf{T}_2:
\end{array}
\left(\begin{array}{cccccccccc}
\epsilon & \epsilon & \epsilon & \overline{\mathsf{l}(\mathsf{p},\mathsf{a}_1)} & \mathsf{f}(\mathsf{a}_1,\mathsf{a}_2) & \overline{\mathsf{u}(\mathsf{p},\mathsf{a}_2)} & \epsilon & \epsilon & \epsilon \\
\mathsf{l}(\mathsf{p},\mathsf{d}_1) & \mathsf{m}(\mathsf{d}_1,\mathsf{a}_1) & \overline{\mathsf{u}(\mathsf{p},\mathsf{a}_1)} & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
\mathsf{m}(\mathsf{d}_2,\mathsf{a}_2) & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \overline{\mathsf{l}(\mathsf{p},\mathsf{a}_2)} & \mathsf{m}(\mathsf{a}_2,\mathsf{d}_2) & \mathsf{u}(\mathsf{p},\mathsf{d}_2) \\
8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 &
\end{array}\right)
$$

Figure 6: A multiagent plan solving the initial LOGISTICS problem used in the experiments. Empty actions are denoted as ϵ. The overlines mark public actions. The numbers in the last row represent particular counts of steps, i.e., number of actions $|\pi| - k$, to the end of the plan.

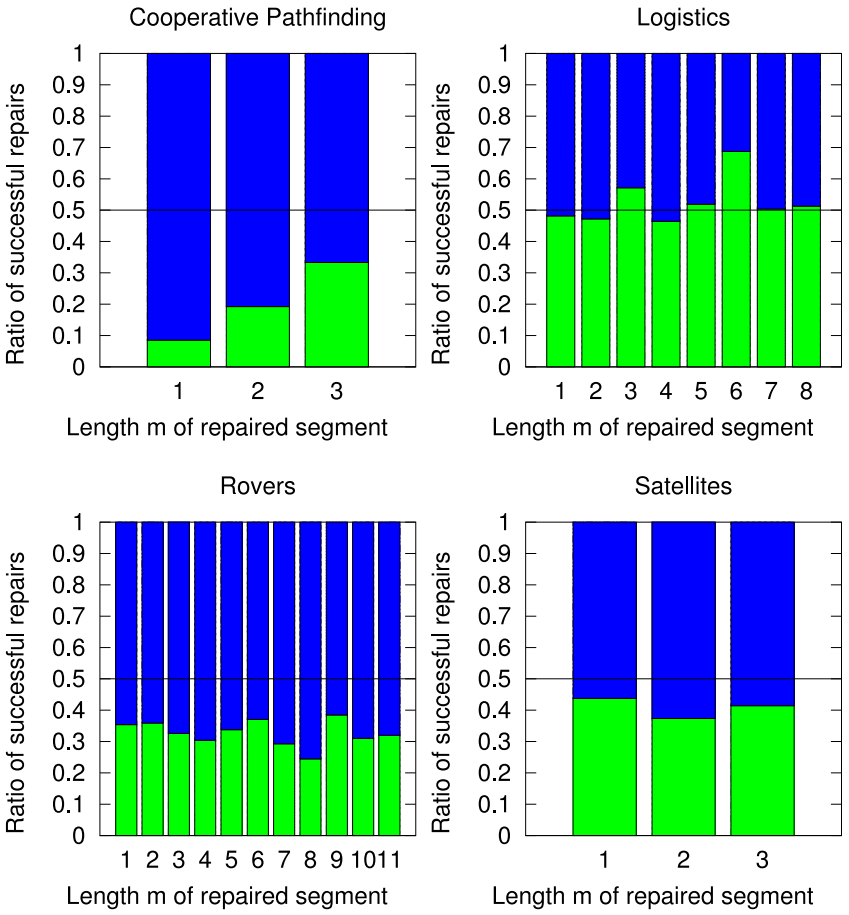


Figure 7: Comparison of success ratio against replanning between suffix preserving (green, back-on-track) and prefix preserving (blue, lazy) plan repairing with variable length $m = |\pi| - k$ of the repaired plan segment.

count of actions in the plan. In SATELLITES, all actions are private and therefore actions of one agent depend only on other actions of the same agent. The highest dependency metrics is in such case $\max_{a \in \bigcup_{i=1}^{n} A_i} \mathrm{dep}_{\pi}(a) = |\pi|$ for the first action a of each satellite in the multiagent plan π. The individual plans of the agents are relatively short (three to four actions) and therefore the dependency metrics is never higher than four and it is zero for public projection π^{pub} of the plan π.

Multiagent plans for the ROVER problems contain several public actions at the end of the plan, representing always only one rover communicating at one time point. Although the plans solving the ROVERS problems contain public actions, there are again no long dependencies among the actions. The dependencies in the private part of the plan contain three components, each containing three to four private actions. Consequently, the private dependencies are, similarly to the SATELLITE problems, maximally four actions long. The dependencies among the public actions are even shorter, as there is the same number of public actions as agents, which means maximally three-action public dependencies for three agents. The dependency link between one public action and one dependent private component increases the maximal dependent length to maximally seven actions (four private actions of the component bound to three public actions successively dependent on each other). Using the dependency metrics $\max_{a \in \bigcup_{i=1}^{n} A_i} \mathrm{dep}_{\pi}(a) = 4+n$ and $\max_{a \in \bigcup_{i=1}^{n} A_i} \mathrm{dep}_{\pi^{\mathrm{pub}}}(a) = n$ for the public plan and for n rovers.

In such repair problem, even if one of the leading actions in a private component fail, prefix preserving (i.e., lazy) approach solves nearly the complete problem only by reusing the original plan. More precisely, it reuses the original solution for the rest of the private components and all the public actions except one of the failed agent. As the results show, the prefix-based repair is always better then the suffix-based and the ratio between these two is rather stable over different points in the plan.

The situation changes in the LOGISTICS domain. In LOGISTIC with three agents and one package, there is a chain of dependent actions. Particularly, $\mathsf{u}(\mathsf{p}, \mathsf{d}_2)$ depends on $\overline{\mathsf{l}(\mathsf{p}, \mathsf{a}_2)}$, which depends on $\overline{\mathsf{u}(\mathsf{p}, \mathsf{a}_2)}$ and so on to the first action of the plan $\mathsf{l}(\mathsf{p}, \mathsf{d}_1)$. The dependency chain has six public actions in the example plan and occupy the complete length of it. As the results show in Figure 7, there are two distinctive peaks where the suffix preserving repair outperforms the prefix preserving repair, additionally with a increasing trend. The first one is for repair plans of length $m = 3$ and the other one is for $m = 6$. As presented in Figure 6, these lengths correspond to the package handover points in the plan, more precisely, to repair of failing unloads $\overline{\mathsf{u}(\mathsf{p}, \mathsf{a}_1)}$ and $\overline{\mathsf{u}(\mathsf{p}, \mathsf{a}_2)}$. These public actions' dependency metrics is $\max_{a \in \bigcup_{i=1}^{n} A_i} \mathrm{dep}_{\pi^{\mathrm{pub}}}(a) = \frac{2|\pi^{\mathrm{pub}}|}{3}$, which in contrast to ROVERS is dependent not only on the number of the agents, but on the length of the public plan. Ignoring a failure of unloading by the prefix preserving (i.e., lazy) approach causes the package is left in the last vehicle and the rest of the team finishes the executable remainder of the plan, which in principle means the vehicles are moving, but they are not transporting the package. On the other hand, in the same circumstances, the suffix preserving repair (i.e., back-on-track) only repeats the unload action and successfully continues with the rest of the original plan ending in a goal state.

One can argue that the complement load actions should be repaired more efficiently

using this same argumentation as well. This is very true, however this phenomenon is not captured in the results, because of a particular implementation of the MA-Plan planner. The explanation is based on the fact the used planner efficiency is more dependent on small differences in number of involved agents, than the number of planned actions. In the case of $m = 3$ (the $\overline{\mathsf{u}(\mathsf{p}, \mathsf{a}_2)}$ action), 2 agents are needed to do lazy repair, because firstly the executable remainder of the original plan is reused to the last state without the package and than the planner has to be used to generate repair plan π_{fix} reverting all the moves and planning to one of the goal states again. Such plan has to firstly unload the package from the airplane A and then transport it successfully by the truck T_2 to the goal destination d_2. On the other hand, the pure suffix preserving approach generates only a plan repeating the unload action $\overline{\mathsf{u}(\mathsf{p}, \mathsf{a}_2)}$ and afterward continues with the original plan as a suffix. This planning problem involves only one agent, in particular, the airplane A carrying out unload of the package. The same principle can be applied to $m = 6$, but with all three agents for pure preserving (lazy) repair, but only 2 agents for pure suffix repair (back-on-track).

In the last problem of COOPERATIVE PATHFINDING, the length of a sequence of dependent actions correspond to the length of the plan as well, as all the actions in such plan are public and inter-dependent. Nevertheless, this is quite different "order of dependency", than in SATELLITES for example. In SATTELITES, all the actions are dependent as well, but only within one agent, whereas here, the actions are dependent across the agents. In the experimental results of the COOPERATIVE PATHFINDING a trend arises. In such dense types of inter-dependent problems, the longer are the repaired plans, the more the suffix repair algorithm gains against the prefix one.

The results of these experiments, namely of LOGISTICS and COOPERATIVE PATHFINDING, *moderately support the third hypothesis* of the paper.

5.3 Partitioning of the Original Plan

It is not intuitively clear what is a good strategy for reusing the original plan parts, moreover related to a particular planning domain. The experiments conducted in these sections provide several insights into this issue and focus especially on answering the Hypothesis 4.

5.3.1 Used Plan Repair

A battery of plan repairing strategies was prepared to validate Hypothesis 4. We parameterize how much generalized repair reuse the original plan. Such parameterization (based on the U and V indice vectors) lead to a two-dimensional discrete space of different plan repairing strategies, as depicted in Figure 8, representing a structure of the repaired plan.

Each of the nine diagrams in the figure describes a variation on a resulting plan repaired by one particular parameterization of the algorithm in the context of execution of the original plan. The execution starts with a world in the initial state I and it is anticipated to continue with help of the original plan to the last state S_m, which is one of the goal states, i.e., $S_m \supseteq G$. However during execution of an action following a state S_k, execution failed and the state of the world ends up not in the state S_{k+1}, but

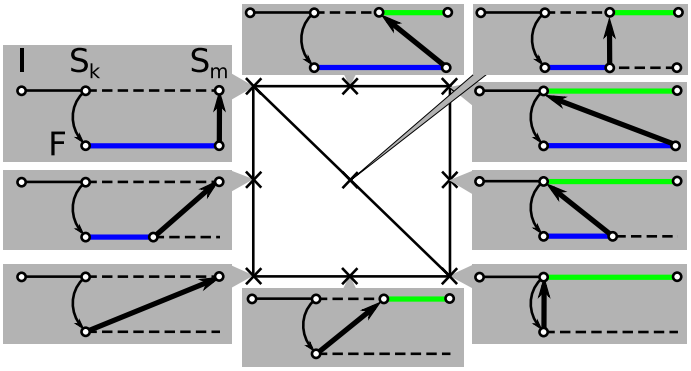


Figure 8: Scheme of a two-dimensional space representing plan repairing strategies preserving different parts of the original plan and reusing it in different ways. The blue segments represent prefix re-usage and the green ones the suffix re-usage. The notable states are: initial state I, last achieved state S_k induced by the original plan, exceptional state F after a failure and the last anticipated state $S_m \supseteq G$, provided that the original plan would be executed without a failure.

in a state F, out of the anticipated sequence of states and actions. To fulfill the goal, the agents use one of the plan repairing strategies, which under the condition of perfect execution, would transform the world from F to a $S_m \supseteq G$.

In Figure 8, there are two dimensions depicted. One of the dimensions represent the number of actions which has to be reused from beginning of the original plan as a prefix corresponding to fixation of the iteration parameter $U = (|\pi| - k)$. The other dimension represents number of actions re-used as suffix of the final repairing plan, i.e., fixed iteration parameter $V = (|\pi| - k)$. In the presented scheme, π_{pre} from the Algorithm 3 is denoted as a blue line, π_{suf} as a green line and π_{fix}as a black thick arrow. Since both the dimensions reuse the same original plan, the space is always a square with a side of the length $|\pi| - k$.

There are four extremes in the repair strategy space. The strategy at position $(0, 0)$ effectively degenerates from $\pi_{\mathrm{pre}} \cdot \pi_{\mathrm{fix}} \cdot \pi_{\mathrm{suf}}$ to π_{fix}. Such process correspond to replanning from the scratch. The strategies at positions $(|\pi| - k, 0)$ and $(0, |\pi| - k)$ represent pure repairs $\pi_{\mathrm{pre}} \cdot \pi_{\mathrm{fix}}$ and $\pi_{\mathrm{fix}} \cdot \pi_{\mathrm{suf}}$ respectively. The last extreme at $(|\pi| - k, |\pi| - k)$ represent an strategy, which firstly uses the original plan ignoring inapplicable actions, then using a newly generated plan π_{fix} returns to the anticipated state after execution of the failed action S_k and than it reuses the original plan again to get to the goal state, i.e., the algorithm generates a full overlap of the prefix and suffix plans.

Beside the extremes, also the $(0, m), (1, m-1), ..., (m-1, 1),(m, 0)$ diagonal for $m = |\pi| - k$ in the space is important from perspective of the ongoing discussion. All the strategies lying on this diagonal can re-use all the actions of the original plan exactly once and in the original order. Meaning, the original plan is neither overused nor underused. Formally, we define:

Definition 12. (m-*normal generalized plan repair*) Let $\Sigma = \langle \Pi, \pi, F, k \rangle$ be a multia-

gent plan repairing problem, then an algorithm R is a m-*normal generalized plan repair*, iff R solves the problem Σ by a multiagent plan π' with decomposition $\pi_{\mathrm{pre}} \cdot \pi_{\mathrm{fix}} \cdot \pi_{\mathrm{suf}}$ and at the same time $(|\pi_{\mathrm{pre}}|, |\pi_{\mathrm{suf}}|) \in \{(0, m), (1, m-1), ..., (m-1, 1), (m, 0)\}$, where $m = |\pi| - k$.

5.3.2 Results and Discussion

To validate the third and last hypothesis, we used a randomized sampling of the strategy space and searched for more successful algorithms lying on the m-normal diagonal. The results are present in Figure 9.

The sampling experimental process measured for each encountered repairing problem the computation time of the replanning strategy. After this base-line measurement, a tested repairing strategy was run with a bound on the computation time based on the replanning run-time. If the strategy performed better, a cell in the result map was incremented by one. In effect, this process rendered the presented normalized results. During the experimental execution and plan repairing, we used different lengths of the original plan, i.e., the repair was done for various $|\pi| - k$. Therefore, the resulting maps depict a continuous space, as the results with higher and lower $|\pi| - k$ values were merged into the most representative m value corresponding to the initial multiagent plan generated.

As the maps show, the hypothesis clearly holds for coupled domains with longer plans (LOGISTICS, and ROVERS). In the coupled domain of COOPERATIVE PATHFINDING, the diagonal is also present, but because of shorter repaired plans, it degenerated considerably. In the experiment with SATELLITES, the diagonal is not present.

These results *support* Hypothesis 2 with an auxiliary observation, that the effect is decreasing as the coupling of the domain decreases.

6 Related Work

There are several approaches capable to drive multiagent team activities in an environment with uncertain dynamics.

Firstly, there is a body of literature dealing with and extending models of decentralized partially observable Markov Decision Processes (Dec-POMDPs) [1]. A Dec-POMDP model leads to computation of a *policy* for the agents in the environment ensuring that by following it, the team reaches (joint) rewards. The model assumes only partial observability of the environment and it is capable of capturing various eventualities which can occur in the environment. The eventualities, however, have to be probabilistically known *a priori*, such that a model of action outcomes can be constructed before planning. Dec-POMDP solvers do not scale well to larger problems, especially when the model of run-time action failures is *a priori* unknown. The plan repairing algorithms proposed in this article do not require an explicit failure model as an input. The price the plan repairing algorithms pay is their inefficiency in problems with failures taking the system far from the presumed evolution based on the original plan and rate of such failures as shown in [10]. A simplified single-agent models described by Markov Decision Processes (MDP) were tackled by scalable techniques of online

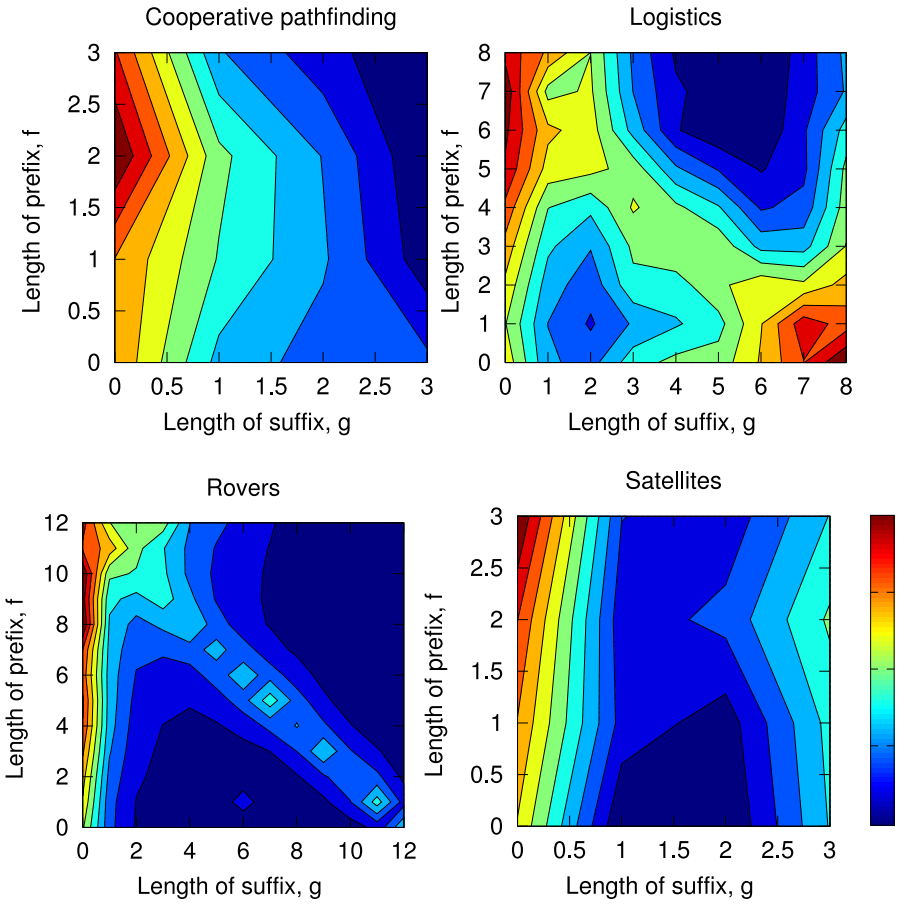


Figure 9: The maps present prefix (u on y-axis) vs. suffix (v on x-axis) preserving repairing algorithms by a success rate against replanning in the repair time for all domains with three agents and $p = 0.3$. Red color represents repair strategies faster then replanning. The top-left to bottom-right diagonal represent algorithms neither overusing or underusing the original plan.

policy replanning by problem determinization e.g., in FF-Replan [16]. Determinization approaches are related to the online replanning scheme. Since we do not assume the probabilistic model of the failures for the multiagent plan repair algorithms, we cannot prepare the (partial) policies using the determinization of the model. Our approach focuses on efficiency of the replanning process per se, when it is needed because of a failure. In other words, plan repair assumes an dynamic and optimistic determinization of the problem based on the original plan.

Secondly, single-agent Contingency [11], Fault Tolerant [7] and Conformant [14] planning techniques facilitate classical-style planning for domains with non-probabilistic uncertainty in either action outcomes or state the system happens to be in. However, again, in order to plan for actions in such domains, the possible contingencies and action models in the environment must be known before the planning phase. In the multiagent plan repair, it is assumed any possible outcome of an action in general. In Contingency, Fault Tolerant, or Conformant planning such assumption would lead to actions possibly taking the environment to any state. Thus a complete graph representing all transitions would render the techniques unusable.

Lastly, the idea of macro actions used for single-agent plan repair [15] build upon the positive results of planning with prescribed sequences of primitive actions. The technique stemmed from the area of integrating planning and machine learning [4] and was adapted to describe parts of the repaired plan by fixed macro actions. In respect to the recency of the techniques, it is not surprising that it was not extended for multiagent planning yet.

7 Conclusion

Based on the theoretical and experimental results, we can come up with a summary of heuristic approaches in form of simply usable advices decreasing computation and/or communication overheads during repairing of multiagent plans. These advices can be used for various plan repairing approaches targeting systems with planning agents reusing the original plan in form of combination of prefix and suffix as we proposed in the generalized repair. The results were verified for plan repairing techniques utilizing preservation of the original plan and using an CSP-based multiagent planner to fill prospective discontinuities in the repairing plan. The advices are:

1. *Use plan reuse based on generalized repair only with plans of polynomially bounded length.*
2. *Prefer smaller numbers of involved agents in the plan repairing process.*
3. *Prefer prefix preserving repairing techniques when repairing failures with long dependencies among different agents.*
4. *Prefer m-normal generalized plan repairing algorithms.*

This work opens several interesting questions left for the future work. Most notably, how would another implementation of the underlying multiagent planner affect the results and would it be possible to integrate principles from single-agent search effort estimation approaches, e.g., as in [6] to provide more precise hints how to repair during the execution and repairing process.

Acknowledgments

This research was partly supported by the Czech Science Foundation (grant no. 13-22125S) and partly by USAF EOARD (grant no. FA8655-12-1-2096).

Bios

Antonín Komenda is a research fellow at Agent Technology Center of Czech Technical University (CTU) in Prague since 2007. He holds PhD from Artificial Intelligence and Biocybernetics from CTU with thesis on Domain-independent Multiagent Plan Repair. In his research, he focuses on theory, methods and algorithms for automated multiagent planning, distributed problem solving, coordination and robust multi-agent planning. In 2010, Antonin undertook a 5-week internship at Drexel University in Philadelphia in the group of prof. Regli and in 2013–2014, he was a post-doctoral fellow at Technion – Israel Institute of Technology in Haifa in the group of prof. Domshlak.

Peter Novák is a post-doctoral research fellow in the Algorithmics group at the Delft University of Technology in the Netherlands, in the group of Cees Witteveen since March 2012. Before moving to Delft, he spent more than 2 years as a post-doctoral research fellow in the Agent Technology Center (ATG) of the Czech Technical University in Prague, in the group of Michal Pěchouček. Peter holds the doctoral degree in computer science from Clausthal University of Technology, where he worked in the Computational Intelligence Group under supervision of Jürgen Dix. Peter's long-term research interests are rooted in investigation of interactions of agents with dynamic environments; more specifically, cognitive robotics, (multi-)agent oriented programming of cognitive, or knowledge-intensive agents and multi-agent planning and plan-repair with a strong interest in practical applications of agent-oriented technologies.

Michal Pěchouček is a full professor at the Czech Technical University in Prague, Head of Agent Technology Center, Deputy head for research at Department of Computer Science, Director of Open Informatics Programme. In 2011 visiting professor at University of Southern California, TEAMCORE lab (Fulbright fellow), in 2006 visiting scientist at University of Edinburgh (Royal Society of Edinburgh), in 2003 visiting professor at State University of New York at Binghamton, in 2000 postdoc researcher at University of Calgary. Chair of the board of directors for EURAMAS, member of the board of directors of IFAAMAS. Visiting/honorary member of the Artificial Intelligence Application Institute, University of Edinburgh, member of the advisory board of the Center for Advanced Information Technologies, SUNY Binghamton. Member of the scientific council at Masaryk University, School of Informatics, Member of Czech Science Foundation Panel on Computer Science.

References

[1] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Math. Oper. Res.*, 27(4):819–840, November 2002. http://dx.doi.org/10.1287/moor.27.4.819.297.

[2] Ronen I. Brafman and Carmel Domshlak. From one to many: Planning for loosely coupled multi-agent systems. In *Proceedings of ICAPS*, pages 28–35, 2008. http://www.aaai.org/Library/ICAPS/2008/icaps08-004.php.

[3] Steve Chien, Minh Binh Do, Alan Fern, and Wheeler Ruml, editors. *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*. AAAI, 2014. http://www.aaai.org/Library/ICAPS/icaps14contents.php.

[4] Lukás Chrpa, Mauro Vallati, and Hugh Osborne. Learnability of specific structural patterns of planning problems. In *ICTAI*, pages 18–23. IEEE, 2013. http://dx.doi.org/10.1109/ICTAI.2013.14.

[5] Rina Dechter. *Constraint processing*. Elsevier Morgan Kaufmann, 2003. http://www.elsevier.com/wps/find/bookdescription.agents/678024/description.

[6] Chris Fawcett, Mauro Vallati, Frank Hutter, Jörg Hoffmann, Holger H. Hoos, and Kevin Leyton-Brown. Improved features for runtime prediction of domain-independent planners. In Chien et al. [3]. http://www.aaai.org/ocs/index.php/ICAPS/ICAPS14/paper/view/7939.

[7] R. M. Jensen, M. M. Veloso, and R. E. Bryant. Fault tolerant planning: Toward probabilistic uncertainty models in symbolic non-deterministic planning. In *ICAPS*, pages 335–344, 2004. http://www.aaai.org/Library/ICAPS/2004/icaps04-040.php.

[8] Antonín Komenda, Peter Novák, and Michal Pěchouček. Decentralized multi-agent plan repair in dynamic environments (Extended Abstract). In *Proceedings of AAMAS*, pages 1239–1240, 2012. http://dl.acm.org/citation.cfm?id=2343941&dl=ACM&coll=DL&CFID=612050604&CFTOKEN=59981303.

[9] Antonín Komenda, Peter Novák, and Michal Pěchouček. How to repair multi-agent plans: Experimental approach. In *Proceedings of Distributed and Multi-agent Planning (DMAP) Workshop of 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, 2013. http://icaps13.icaps-conference.org/wp-content/uploads/2013/05/dmap13-proceedings.pdf.

[10] Antonín Komenda, Peter Novák, and Michal Pěchouček. Domain-independent multi-agent plan repair. *Journal of Network and Computer Applications*, 2013. http://www.sciencedirect.com/science/article/pii/S1084804512002585.

[11] Christian J. Muise, Vaishak Belle, and Sheila A. McIlraith. Computing contingent plans via fully observable non-deterministic planning. In Carla E. Brodley and Peter Stone, editors, *AAAI*, pages 2322–2329. AAAI Press, 2014. http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8656.

[12] B. Nebel and J. Koehler. Plan reuse versus plan generation: a theoretical and empirical analysis. *Artificial Intelligence*, 76(1-2):427–454, July 1995. http://www.sciencedirect.com/science/article/pii/000437029400082C.

[13] Raz Nissim, Ronen I. Brafman, and Carmel Domshlak. A general, fully distributed multi-agent planning algorithm. In *Proceedings of AAMAS*, pages 1323–1330, 2010. http://dl.acm.org/citation.cfm?id=1838379.

[14] Hector Palacios and Hector Geffner. Compiling uncertainty away in conformant planning problems with bounded width. *J. Artif. Int. Res.*, 35(1):623–675, August 2009. http://dl.acm.org/citation.cfm?id=1641503.1641518.

[15] Enrico Scala. Plan repair for resource constrained tasks via numeric macro actions. In Chien et al. [3]. http://www.aaai.org/Library/ICAPS/icaps14contents.php.

[16] Sung Wook Yoon, Alan Fern, and Robert Givan. Ff-replan: A baseline for probabilistic planning. In Mark S. Boddy, Maria Fox, and Sylvie Thiébaux, editors, *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007, Providence, Rhode Island, USA, September 22-26, 2007*, page 352. AAAI, 2007. http://www.aaai.org/Library/ICAPS/2007/icaps07-045.php.