Domain-independent Repairing of Multiagent Plans (Dissertation Abstract)

Antonín Komenda
(antonin.komenda@agents.fel.cvut.cz)

Agent Technology Center,
Department of Computer Science and Engineering, Faculty of Electrical Engineering,
Czech Technical University in Prague, Czech Republic

Achieving joint objectives in distributed domain-independent planning problems by teams of cooperative agents requires significant coordination and communication efforts. For systems facing a plan failure in a dynamic environment, arguably, attempts to repair the failed plan in general, and especially in the worst-case scenarios, do not straightforwardly bring any benefit in terms of time complexity. However, in multiagent settings, the communication complexity might be of a much higher importance, possibly a high communication overhead might be even prohibitive in certain domains. The hypothesis is that in decentralized systems, where frequent coordination is required to achieve joint objectives, attempts to repair failed multiagent plans should lead to lower communication overhead than replanning from scratch.

The key distinction of domain-independent multiagent planning as defined by (Brafman and Domshlak 2008) is precise separation of *individual* and *public* knowledge of the agents during planning. That means, actions and facts describing the environment in the planning process can be known only to one particular agent making them effectively private or known by more than one agent making them public. The distinction is similar to the shift in the research from centralized constraint satisfaction problems (CSPs) to their distributed version DisCSPs also based on a similar separation scheme.

The individual and public distinction as defined in (Brafman and Domshlak 2008) has another additional benefit in possible simplification of the planning problem for loosely coupled domains. Such plan factorization approaches are not new in planning as a technique for simplification of several planning problems (Brafman and Domshlak 2006). The specifics of the factorization in the multiagent planning is in the way, how is the factorization done. In the factorization approaches in classical planning the key variable is how is the problem decomposed into factors, however in the multiagent problems, the factorization is naturally defined by the abilities of the agents, which dictate precisely how the factorization has to be done. The price for this particular scheme is a decreased generality of the decomposition.

Being agents in dynamic and uncertain environments without an unknown failure model, actions and plans may not always have desired consequences and moreover, there is no a priori knowledge about the the failures, as is required for MDP-based approaches. To account for such failures, a cautious agent must be able not only to execute its actions, but also monitor its own progress, detect failures and prospectively change its own plan to reflect the unforeseen phenomenons. Generally speaking, beside a planning component, the agent should also include a monitoring and a replanning or plan repairing component. More concretely, considering a multiagent plan produced by a suitable multiagent planner, the abstract execute-monitor algorithm checks in every state the soundness of the current plan for next step before advancing. If necessary, it invokes a fixing procedure.

Creating an effective plan repairing procedure has its limitations and caveats as Nebel and Koehler showed in (Nebel and Koehler 1995). An approach to minimize such risks was to create plan repairing algorithms based on any available multiagent planner and utilize its positive properties for the good of the repairer as well. The plan repairing algorithms in the thesis are all build on a state-of-the-art multiagent planner by (Nissim, Brafman, and Domshlak 2010) based on compilation of the public part of a multiagent planning problem into a DisCSP problem. For the individual parts, it is used a forward-chaining A* based state-space search planner, particularly FastForward. This planner was used both as the planner for the initial multiagent plans and as the planning component in the repairing algorithms. The proposed plan repairing algorithms are based on the principles of the MODDELINS modifications (Nebel and Koehler 1995) and bring them into the multiagent setting as presented in (Komenda, Novák, and Pěchouček 2013).

The core idea behind the first plan repairing approach coined *back-on-track (BoT)* is to utilize a multiagent planner to produce a plan from the failed state to the originally desired state and subsequently follow the rest of the original multiagent plan from the step in which

the failure occurred. In result, the BoT repair tries to preserve a suffix of the original plan and prefix it with a newly computed plan starting in the failure state and leading to some state along the execution of the original plan in the ideal environment.

The second approach, *lazy-repair (LR)*, is designed to preserve an *executable remainder* of the original multiagent plan (remain only actions, if the original plan was executed ignoring non executable actions) and close the gap between the state resulting from the failed plan execution and a goal state of the original planning problem. The lazy approach tries to preserve a partial prefix of the original plan and complete it by a newly planned plan suffix. The algorithm is incomplete, as it might happen that the execution of the executable remainder diverges to a state from which no plan to some goal state exists.

The shortcoming of the LR algorithm is addressed by the *repeated lazy repair (RL)*. The idea is that a failure during execution of an already repaired plan makes the previous repair irrelevant and its result can be discarded, unless the failure occurred already in the fragment appended by the previous repair. Note, the repeated lazy repair algorithms enables a plan execution model which preserves significantly longer fragments of the original plan. That is, upon a failure, instead of trying to repair the failed plan directly, as the previous two algorithms, the system can simply proceed with execution of the remainder of the original plan and only after its complete execution the lazy plan repair is triggered. The approach simply ignores the plan failures during the multiagent plan execution and postpones the repair to the very end of the process, hence the "*lazy*" label for the two algorithms.

The last plan repairing algorithm is a generalization of the BoT and LR approaches and it is designed to inspect various combinations of the prefix and suffix plan preservation approaches.

The plan repairing algorithms were tested in a spectrum of experiments validating both computational and communication complexity, length of the resulting repaired plans and numbers of messages passed in the system. The used planning domains were multiagent extensions of classical IPC[1] benchmarks, e.g., LOGISTICS, ROVERS, SATELLITES and others. The domains were enriched by two particular types of plan failures: *action failures* and *state perturbations*. Both failure types were parametrized by an uniformly distributed probability P, which determines whether a simulation step fails, or not. Both failure types are weak failures. That is, they are not handled immediately, but can preclude the plan execution and later result in a strong failure, i.e., inapplicability of an action. Upon detection, a strong failure is handled by one of the plan repairing algorithms. An action failure is simulated by not-execution of some of the individual agent actions from the actual plan step. The individual action is chosen according to a uniform probability distribution over the positions within a joint action. The individual failed action is then removed from the joint action and the current state is updated by the modified joint action. The other simulated failure type, state perturbation, is parametrized by a positive non-zero integer c, which determines the number of state terms, which are removed from the current state, as well as the number of terms which are added to it. The terms to be added or removed are selected also randomly from the domain language according to a uniform distribution.

The plan repairing techniques were also adapted and verified in simulation environment of tactical missions. The dissertation presents a simulation-aided development process and a related software toolkit for iterative adaptation of the plan repairing algorithms from synthetic environments to a high-fidelity simulation. The multiagent plan repair is needed to amend plan activities for a team of agents in a highly dynamic environment of the tactical missions. The solution is based on the Back-on-Track plan repairing algorithm preserving suffix part of the original plans. The adaptation of the algorithm for the domain of tactical support led to an introduction of a restricting condition on the depth of the search tree to limit the computational complexity of the search. The plan-repairing mechanism also addresses the problems caused by the uncertain movement of the troops in the dynamic environment.

The work is concluded by a summary of perspective open challenges for future work. Firstly, used multiagent planning framework is not expressive enough to describe certain aspects of concurrent actions and should be extended to this end. Secondly, there is a need for more efficient and feature-full implementations of multiagent planners, as the gap between the state-of-the-art classical planners and multiagent planners is enormous. Thirdly, there is a lack of standardized planning benchmarks for multiagent planning, especially considering tightly coordinated planning problems.

[1] International Planning Competition

References

Brafman, R. I., and Domshlak, C. 2006. Factored planning: How, when, and when not. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-2006)*, 809–814.

Brafman, R. I., and Domshlak, C. 2008. From one to many: Planning for loosely coupled multi-agent systems. In *Proceedings of ICAPS*, 28–35.

Komenda, A.; Novák, P.; and Pěchouček, M. 2013. Domain-independent multi-agent plan repair. *Journal of Network and Computer Applications*. DOI: 10.1016/j.jnca.2012.12.011.

Nebel, B., and Koehler, J. 1995. Plan reuse versus plan generation: a theoretical and empirical analysis. *Artificial Intelligence* 76(1-2):427–454.

Nissim, R.; Brafman, R. I.; and Domshlak, C. 2010. A general, fully distributed multi-agent planning algorithm. In *Proceedings of AAMAS*, 1323–1330.