Multi-Agent Planning with Mixed-Integer Programming and Adaptive Interaction Constraint Generation (Extended Abstract)

Jan-P. Calliess and Stephen J. Roberts

Abstract

We consider multi-agent planning in which the agents' optimal plans are solutions to mixed-integer programs (MIP) that are coupled via integer constraints. While in principle, one could find the joint solution by combining the separate problems into one large joint centralized MIP, this approach rapidly becomes intractable for growing numbers of agents and large problem domains. To address this issue, we propose an iterative approach that combines conflict detection with constraint-generation whereby the agents plan repeatedly until all conflicts are resolved. In each planning iteration, the agents plan with as few other agents and interaction-constraints as possible. This yields an optimal method that can reduce computation markedly. We test our approach in the context of multi-agent collision avoidance in graphs with indivisible flows. Our initial simulations on randomized graph routing problems confirm predicted optimality and reduced computational effort.

Introduction. Mixed-integer programming (*MIP*) methods are a general tool for solving (multi-agent) planning problems. Indeed, since MIP algorithms can solve the NP-complete multi-commodity flow problem (Even *et al.* 1976), by reduction, they harbour the potential for solving any NP problem. Unfortunately, this universality property implies that, unless P=NP, there is no hope of guaranteeing tractable performance on all problem instances. Nonetheless, the promise of optimality of a MIP solver's output upon termination has spawned an abundance of papers proposing MIP to be applied in various settings, including model-predictive control, sensor networks and graph routing with indivisible flows both in single- and multi-agent scenarios (e.g. (Schouwenaars *et al.* 2001; Calliess *et al.* 2011; Blackmore *et al.* 2010)). While the availability of highly optimized MIP libraries allows such approaches to exhibit good average-case performance in low-dimensional applications with a limited number of constraints, ordinarily MIPs scale poorly in the number of constraints and dimensionality of the domain. The latter is exacerbated in multi-agent scenarios where the dimensionality scales linearly in the number of agents. This effectively denies application to larger agent collectives. To address such tractability concerns, an abundance of heuristic aproaches have been proposed, including methods based on Fixed-Priorities (FP) (Erdmann & Lozano-Perez 1987) and auctions. Unfortunately, these normally yield suboptimal solutions and often are not guaranteed to terminate.

In contrast, we propose two approaches that are based on adaptive constraint generation to ameliorate tractability. The first, *CCG*, aims to reduce the number of interaction constraints of a centralized MIP to generate the optimal joint solution. In addition, the second method, SDCG, also attempts to reduce the dimensionality of the domain by uncovering a decomposition into cliques of independent subproblems. For both methods, we assume all agent interactions are modelled as constraints. We call solutions violating these constraints to be in *conflict*. For instance, in the graph routing scenario considered below, such constraints will prohibit collisions – i.e. no node is allowed to be used by two agents simultaneously.

Proposed methods. *CCG*: The approach aims to solve the centralized MIP for the joint solution with a small number of interaction constraints. In an iterative process, CCG solves a sequence of centralized MIPs. Each MIP jointly optimises the agents' solutions simultaneously in (high-dimensional) joint planning space. The initial MIP contains no interaction constraints. After a joint solution is generated it is examined for conflicts. Any interaction constraint that is violated is then added to the definition of the MIP before re-planning begins. The whole process is repeated until a conflict-free joint solution is found.

SDCG: At the outset, each agent solves its own (low-dimensional) MIP independently, without any interaction constraints. The agents then exchange their solutions and detect *conflicts*. Conflicting agents merge into clusters. Within each cluster, they plan centrally by combining their MIPs into one larger joint MIP that is solved with our CCG method. That is, each of these cluster MIPs contains the constraints of the individual MIPs as well as exactly those interaction constraints that suffice to prohibit the previously detected conflicts among the agents within the cluster. Whenever solutions of different clusters emerge to be in conflict, the corresponding clusters are merged. The whole process of merging and re-planning repeats until all conflicts are resolved.

Experiments. We apply our methods to multi-agent trajectory planning in graph environments. Incurring edge-

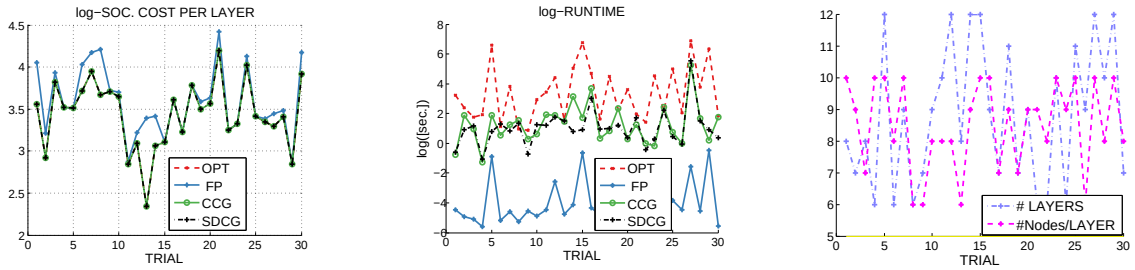

Figure 1: Left: Joint costs. As expected, the optimal joint costs (OPT) coincide with those incurred by CCG and SDCG. Centre: runtime over 30 randomized graph routing problems with 5 agents and varying graph sizes, time horizons and edge-transition costs. Curves show log-scale performances of full centralized MIP solvers without constraint generation (OPT), the Fixed-Priority method (FP) and our CCG and SDCG methods. Right: number of spatial nodes and temporal layers (time steps) in each trial.

transition costs along the way, agents can move from node to node along edges. Each agent aims to reach a given goal node without colliding with any other agent.

Following a simple unrolling process of the spatial graph into a spatio-temporal graph, the problem can be reduced to the multi-commodity flow problem with atomic flows and unit node-capacities. That is, the trajectory planning problem in the spatial graph is reduced to the path planning problem of finding min-cost disjoint paths connecting goals and start nodes in the corresponding spatio-temporal graph (cf. (Calliess *et al.* 2011)). It is straight-forward to find an optimal joint plan for this problem, i.e. a sequence of edge transitions for the agents, as a solution to a mixed-integer program.

To gain a first impression of the viability of our methods for this task, we conducted preliminary tests on a number randomized graphs in comparison to a fully centralized MIP solver (OPT) (which is known to find the optimal joint plan) and the Fixed-Priority method (FP) (as an example of a heuristic approximation method). Our simulations were executed on a laptop running MATLAB R2013a on a single Intel(R) CORE(TM) i7-2640M @2.8GHz with 8 GB RAM. The solver used was MATLAB's inbuilt *bintprog*. In Fig. 1 the plots depict the relative log-scale performances of the different methods along the metrics of social cost (of the generated plans) and runtime. CCG and SDCG required significantly more computation than the FP heuristic. However, in contrast to FP, they always managed to find the cost-optimal solution while being markedly faster than OPT. Note, on certain trials omitted from the plots, FP did not succeed in finding a feasible solution. By contrast, all three optimal methods did.

Discussion. Our methods have the following properties:

(i) *Optimality.* The proof relies on the fact that in each iteration, a relaxed version of the optimal centralized MIP is solved. In the iterative process, conflict detection then uncovers a superset of those interaction constraints that would be active for an optimal plan vector under the full global MIP.

(ii) *Improved average-case tractability*. One can conceive problem instances where our methods eventually solve the full centralized MIP (OPT). Since computation is expended before this final MIP is created, they would be slower than OPT in such cases. Often however, agent plan interactions occur on a small subset of plan dimensions (between few agents) and resources (e.g. nodes). In such cases, optimal solutions only involve few active interaction constraints that may each involve only a fraction of plan vector components each. For instance, in large graphs where agents will never (or only sparsely) encounter each other, it would be computationally wasteful to generate plans based on mixed-integer problems that model even those potential interactions that will never occur near the optimal solution anyway. Consequently, a lot can be gained by our method in such scenarios. Our initial experiments confirm this intuition.

(iii) *Semi-distributivity (SDCG)*. In cases, where the full centralized MIP does not have to be solved, SDCG may succeed in decomposing the large joint problem into low-dimensional sub-problems that could be solved more locally and in parallel. However, since the approach may eventually lead to the necessity to solve the full central MIP in the worst case, we cannot claim that SDCG is fully distributed.

Ongoing work aims at further elucidating the relative performances of the different methods. For instance, we anticipate parallelized implementation of SDCG to yield superior average run-time performance over CCG. Furthermore, under which statistical assumptions about the graph environment would we expect one method to outperform another? Finally, we aim to develop machine learning algorithms that can predict the best method for a given problem instance.

References

Blackmore, L.; Ono, M.; Bektassov, A.; and Williams, B. 2010. A probabilistic particle approach to optimal, robust predictive control. *IEEE Trans. on Robotics*.

Calliess, J.; Lyons, D.; and Hanebeck, U. 2011. Lazy auctions for multi-robot collision avoidance and motion control under uncertainty. Technical Report PARG-01-11, Dept. of Engineering Science, University of Oxford.

Erdmann, M. A., and Lozano-Perez, T. 1987. On multiple moving objects. *Algorithmica*.

Even, S.; Itai, A.; and Shamir, A. 1976. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing* 5(4):691–703.

Schouwenaars, T.; De Moor, B.; Feron, E.; and How, J. 2001. Mixed integer programming for multi-vehicle path planning. In *European Control Conference*.