Multi-Agent Path Planning with Constraints on Visitation Order

AAAI 2015 Submission 2327

Abstract

We consider path planning problems with multiple cooperating agents, each with its own start and goal position, the optimization target being the last time of arrival. The agents are coupled together by constraints on the order in which certain regions are visited. Such constraints can encode constructive interactions, such as when agents need to complete actions like opening doors for one another, as well as destructive interactions, where one action precludes another thereafter. We show how this class of constraints can describe a variety of practical domains in maintenance, surveillance, and assembly. Next we show that, unlike classical graph search, visitation order constrained planning is NP-hard, regardless of approximation factor. Nonetheless, we present a planner SeqA* which guarantees completeness with an approximation factor equal to the number of agents. Its performance is optimized using a trie-like data structure. Finally, we evaluate the planner's performance experimentally.

Introduction

The simplest motion planning domains, where the only state is an agent's position, reduce to explicit shortest-paths problems on graphs. On the other hand, high-level domain representations such as STRIPS lead to exponential-size graphs, which are NP-hard in general and demand different techniques to incorporate the high-level information encoding the graph's structure. Here we are interested in a middle ground, often occupied by multi-task and/or cooperative multi-agent planning problems. The agents still move between locations on an explicit graph, but some aspect of their history is relevant toward determining which future moves are allowed. If multiple agents plan jointly, they may help or hinder one another.

A simple and natural example is that of robot A pressing a switch that opens a door which robot B must pass through. In this example, imagine either that robot B is incapable of pressing the switch due to physical limitations, or that the switch happens to lie conveniently on robot A's path so that the collaboration results in faster plan execution. Thus, the N-agent problem cannot simply be solved by N independent planners running in parallel. Nor would be want to search with a single planner over the joint state space of all N agents, as that would result in exponential blowup even when there is a small constant number of constraints. However, once the number of constraints grows, exponential blowup seems unavoidable in the worst case, as we will show that even obtaining a bounded suboptimal solution is NP-hard.

Nonetheless, our algorithm SeqA* easily handles small instances, and is also efficient on a class of instances where it is clear from the graph topology that the constraints can only interact in a few specific orders. If the number of constraints is constant, it runs in time polynomial in the number of agents (not quite, TODO).

Related Works (still kind of sloppy, revise)

Multi-agent planning is a growing field of research, as the emergent complexity of multi-agent systems raises the need for more advanced AI in both cooperative and competitive applications (Van der Hoek and Wooldridge 2008). Previous approaches for combining multiple individual plans into one consistent multi-agent plan include (Georgeff 1988) and (De Weerdt, Ter Mors, and Witteveen 2005). Our approach is different in that, by restricting the allowable constraints to a set that we very often find in practice, we are able to use simpler graph search methods in place of logical methods or complex representations such as STRIPS.

We take the well-studied A* planning method (Hart, Nilsson, and Raphael 1968), and extend it to multi-agent systems which interact via a set of constraints. To our knowledge, the particular class of constraints we consider is new to the field of multi-agent path planning. It is essentially a very restricted subset of temporal logic (Gabbay et al. 1994) in which every variable is monotonic: either constant, starts false and at some point shifts to becoming true thereafter, or starts true and shifts to becoming false.

Different constraint formulations have been explored in the context of multi-agent planning. CBS, MA* and variants (Ferner, Wagner, and Choset 2013)(Sharon et al. 2015) adapt the dimensionality of the search to depend only on the number of agents which are actually in conflict. Our approach, in constrast, does all the low-level planning independently, but at the cost of a recombination step to merge the resulting plans. Perhaps the closest prior domain to ours is one

on which the DPC algorithm (Bhattacharya, Likhachev, and Kumar 2010) was applied. Here, multiple tasks need to be distributed among robots with time-parametrized constraints on their maximum distance. Our ordering constraints generalize the task distribution problem, but on the other hand we drop the distance constraints, lending our problem to more discrete methods.

Problem Formulation

A visitation order (VO)-constrained planning problem with N agents and K constraints is formulated in terms of N weighted directed graphs $\{V_i, E_i, start_i, goal_i\}_{i=1}^N$ and K **VO constraints** $\{\Phi_i\}_{i=1}^K$. The vertex set V_i contains the discrete positions accessible to agent i, including the special start and end positions $start_i, goal_i \in V_i$. Each edge $(u, v, w) \in E_i \subset V_i \times V_i \times [0, \infty)$ connects u to v and takes w time units to traverse. When such an edge exists, we may abuse notation and write $c(u, v) = w$. Otherwise, $c(u, v) = \infty$. We say the problem is **undirected** if $c(u, v) = c(v, u)$ for all u, v. Otherwise, it is **directed**.

A **plan** for agent i is a sequence $\pi_i = \langle (v_0, t_0), (v_1, t_1), \ldots, (v_{m_i}, t_{m_i}) \rangle$ such that $v_0 = start_i$, $t_0 \geq 0$, $v_{m_i} = goal_i$, and for all j: $v_j \in V_i$ and $t_j \geq t_{j-1} + c(v_{j-1}, v_j)$. We say that π_i **visits** v_j at **time** t_j. An assignment of plans $\{\pi_i\}_{i=1}^N$ to all N agents is called a **joint plan**, and is **valid** if it **satisfies** all K of the VO constraints. The **cost** of a joint plan is the maximum arrival time $\max_{i=1}^N t_{m_i}$. A planner's objective is to compute a valid joint plan of low cost. If the cost is at most $\epsilon \geq 1$ times that of a minimum-cost valid joint plan, then we say the computed plan is ϵ-optimal.

A VO constraint $\Phi = \langle \Phi^-, \Phi^+, type \rangle$ is described by two vertex sets $\Phi^-, \Phi^+ \subset \cup_i V_i$, along with a **constraint type** describing the relationship Φ imposes on the relative visit times of Φ^- and Φ^+. To understand the constraint types, fix a joint plan $\{\pi_i\}_{i=1}^N$. For any vertex set $S \subset \cup_i V_i$, let $t_{min}(S)$ and $t_{max}(S)$ denote the smallest and largest time at which some π_i visits some vertex in S. The four constraint types are defined by the following inequalities:

- O (open) constraint: $t_{\min}(\Phi^-) \leq t_{\min}(\Phi^+)$
- C (close) constraint: $t_{\max}(\Phi^-) \leq t_{\min}(\Phi^+)$
- R (restore) constraint: $t_{\max}(\Phi^-) \leq t_{\max}(\Phi^+)$
- S (sequence) constraint: $t_{\min}(\Phi^-) \leq t_{\max}(\Phi^+)$

When the corresponding inequality is met, we say that $\{\pi_i\}_{i=1}^N$ satisfies Φ. Intuitively, an O constraint corresponds to a set of doors Φ^+, which start closed but can be permanently opened by triggering any one of the switches in Φ^-. In a C constraint, it is the Φ^- vertices which should be thought of as doors, which are open until permanently barred shut by triggering a switch in Φ^+. In an R constraint, visiting Φ^- is like using an appliance; after the final use, Φ^+ must be visited at least once in order to "clean up". Finally, the S constraint simply dictates that some instance of Φ^- take place before some instance of Φ^+.

Notice that if neither of Φ^+, Φ^- are ever visited, then by the usual definition of $\min, \max$ of empty sets as $\infty, -\infty$ respectively, Φ is satisfied iff it is not of type S. To prevent any ambiguities, we further require that $\Phi_i^+ \cap \Phi_j^- = \emptyset$ for all $i, j = 1, \ldots, K$. That is, each vertex may appear on the left-hand or right-hand side of several constraint inequalities, but not both.

Theorem 1. *Suppose all constraints are of type O or C. Then given an individual plan for each of the N agents, an optimal join plan consistent with the individual plans can be found, if it exists, by greedily moving the agents forward in parallel along their respective plans, pausing when necessary to respect a constraint.*

Corollary 1. *Suppose all constraints are of type C or R. Then an optimal consistent joint plan can be found, if it exists, by applying the greedy algorithm backwards in time starting from the goals.*

Theorem 2. *A multi-agent planning problem involving O, C, R and S constraints can be reduced to one involving only O and C constraints, such that every plan in one problem can be efficiently transformed into an equal-cost path in the other.*

Proof. Sketch (rewrite later): Representing R contraints in terms of Os and Cs: suppose B restores (i.e. cleans after) A. Then create a copy B' of B. Let B' close A, and B' open any agent's goal node. To deal with the boundary condition where neither A nor B are visited, the start node should have a similar copy, reachable at zero cost.

Representing S contraints in terms of Os: suppose (A, B) are a mandated sequence. Then create a copy B' of B. Let A open B', and B' open any agent's goal node. □

Since R and S constraints can be represented in terms of O and C, we consider the former as the only types without loss of generality. This allows us to use the greedy algorithm according to Theorem [X].

Example Applications

In distributed assembly, it is often necessary for one agent to complete its task before another agent can begin the next step. This is easily encoded by a type O constraint. In maintenance, a type R constraint can indicate that certain restoration tasks must follow equipment use in order to return everything to its original state.

Finally, some interesting encodings are possible in security and surveillance. If, say, k agents must scan the same object, this can be encoded by giving each agent k copies of the vertex where the scanning takes place, each opening one of k O doors blocking the goal. Taking one of the copies blocks the agent' s remaining $k - 1$ copies using an O constraint, thus preventing the same agent from being counted twice. We can also require, for instance, that one agent of each type scan the item, by simply giving different agents control over different "doors".

Hardness Results

In unconstrainted single-agent graph planning, the main difficulty stems from the graph being extremely large, often

too large even to store in memory. If the graph were reasonably compact, say with only a million edges, then Dijkstra's algorithm finds the optimal path very efficiently. We might hope that something similar is true in the presence of VO constraints. However, we show here that unless P = NP, the complexity of VO-constrained planning is superpolynomial in K, the number of constraints. This is the case even if there is only one agent, unit edge costs, and we allow an arbitrarily large constant approximation factor. Thus, we may only hope to solve instances with few constraints in the worst case, and more complex instances if we are able to use the structure of realistic problem instances in a particular domain.

Theorem 3. *Determining the existence of a VO-constrained plan is NP-complete, even in the restricted case where there is only one agent and at most one of the following conditions holds:*

- *All constraints have type O.*
- *The graph is undirected.*

Proof. Sketch (rewrite later): Let's say there are m clauses and the variables are $x_1, x_2, \ldots, x_n$. Draw the edges $(start_1, x_1)$, $(start_1, \neg x_1)$, (x_i, x_{i+1}), $(x_i, \neg x_{i+1})$, $(\neg x_i, x_{i+1})$, $(\neg x_i, \neg x_{i+1})$, $(x_n, goal_1)$, $(\neg x_n, \neg goal_1)$. For the other agent, draw the edges $(start_2, c_0)$, $(c_{i-1}, req_{i,j})$, $(req_{i,j}, c_i)$, $(c_m, goal_2)$. Here, $req_{i,j}$ is a door that needs to be opened by the node corresponding to the j'th variable of the i'th clause. □

On the other hand, if both constraints hold, then we are left with a generalization of the Metric Traveling Salesman Problem. Since the latter admits constant-factor approximations in polynomial time (Christofides 1976), we might expect the same to hold here. However, the following theorem trashes such hopes.

Theorem 4. *Fix $\alpha < 1$. It is NP-hard to find an $\alpha ln(K)$-optimal VO-constrained plan, even when ALL of the following conditions hold:*

- *There is only $N = 1$ agent.*
- *All edges have zero or unit cost.*
- *All constraints have type O.*
- *The graph is undirected.*
- *$|\Phi_i^+| = |\Phi_i^-| = 1$ for all i.*
- *Each vertex appears in at most one constraint.*

Proof. This follows from the inapproximability of SUBSET-SUM (Moshkovitz 2012). To reduce from SUBSET-SUM, let the i'th subset be $\{a_{ij}\}_j$. For this subset, form a connected component of zero-cost edges, containing triggers for the doors labeled by $\{a_{ij}\}_j$. The goal is reachable from the start by following a path of zero-cost edges, along which all these doors are blocking the way. Allow this path to branch whenever there are multiple copies of the same door. Finally, connect the start to each of the connected components by a unit-cost edge. □

Approach

Part of the difficulty in planning with these temporal constraints is that the set of available actions does not depend solely on the current position: it also depends on past positions visited by the agent and its collaborators. Perhaps the most obvious approach, then, is to plan over the joint state space. That is, do an A* search over the graph whose vertices correspond to N-tuples as well as a history bitmask documenting which of the sets $\Phi^{\pm}$ have been visited. If we search about V positions per agent, and there are K constraints, then naively this corresponds to a complexity of about $V^N 4^K$, or $V^N 2^K$ once we notice that only Φ^- visits need be remembered for O constraints and Φ^+ visits for C constraints. Due to the above hardness results, we are willing to tolerate the 2^K term in hopes of K being small. However, V^N is certainly not acceptable! While we don't expect such a planner to fully explore the Cartesian product of all the agents' configuration spaces, this still allows far too much repeat work on each individual agent's states.

Another rather naive approach would be to greedily floodfill positions which are not occuped by any C constraints, since O's correspond to doors which start closed, and opening them can never hurt completeness. While this deals with O's with low computational effort, the resulting paths may do substantial wasted work, and so we any semblance of an A* optimality guarantee. Indeed, Theorem [X] shows that we cannot hope for a reasonable suboptimality bound from an efficient greedy approach. Worse, C's cannot be processed greedily. We could choose to close a door or not, nondeterministically, i.e. try one approach, recurse, and then backtrack if it fails. Or we could branch in such cases, exploring both possibilities in a manner that trades dynamically between breadth and depth like A*. If we also treat the O's this way in place of the greedy approach, a bounded suboptimal algorithm starts to take form.

Therefore, we opt for an approach where each agent runs an independent planner, and then the results are joined according to Theorem [X]. Since individual plans may have to be interleaved in unpredictable ways, we save not only a bitmask but a sequence documenting the order in which constraints were triggered en route to the state under consideration. Indeed, we plan over a new notion of "state" which contains both a position and a sequence encoding the relevant history. The complexity of this representation is $K!N$ per agent, or $K!VN$ overall. Since VN is the total size of the configuration spaces, this is essentially linear when the number of constraints is sufficiently small. For convenient, we search backward from the goal. Thus, every time agent i's planner expands a state $(pos, hist)$ such that $pos = start_i$, the corresponding path is compared with paths from the other agents in the greedy manner described above. Unfortunately, this does present a bottleneck in that we're potentially trying up to $(K!)^N$ path tuples, where each agent may try any sequence of constraints.

Analysis (needs rewriting)

$g_{s_1,\ldots,s_N}(s_i)$ is the minimum time by which agent i can reach state s_i, assuming every agent j will go to s_j along

some known path (following $bp(s_j)$, say) and wait there forever. Note that the state space for each agent is the Cartesian product of its graph's vertex set, and the set of partial permutations of constraints.

Define $f(s_1, \ldots, s_N) = \max_i g_{s_1,\ldots,s_N}(s_i) + \epsilon h(s_i)$. This key is impractical to compute over all possible tuples. Further approximations will need to be made. Note that $(s_1, \ldots, s_N)$ range over the goal nodes as well as the $OPEN$ list.

Even if we could compute f, does it guarantee $N\epsilon$ optimality? To prove that, it would suffice to show that we only expand nodes whose individual-agent g-values are ϵ-optimal. However, the logic fails because passing through a door-opening node can result in a sudden drastic decrease in the $g_{s_1,\ldots,s_N}$ value, due to constraints being lifted. To remedy this, instead of defining g by "wait-forever" semantics, we could treat doors as being opened after the partial paths to $(s_1, \ldots, s_N)$, at some lower bound time.

A lower-bound for $f(s_1, \ldots, s_n)$ could be computed as $f(s_j)$ by building a coarse search tree at the level of constraint sequences, where a macro-expansion consists of searching toward each possible next constraint node.

Want to prove that $g(s)$ is ϵ-optimal at expansion (when its jointPriority value is minimal), and thus the joint paths are $N\epsilon$ optimal. Suppose $g(s) > \epsilon g^*(s)$. As induction hypothesis, let $s' \in OPEN$ be such that $g(s') \leq \epsilon g^*(s')$. Fix a priority-minimizing tuple $(s_1, \ldots, s, \ldots, s_n)$ for s. Now consider the tuple $(s_1, \ldots, s', \ldots, s_n)$. If its priority were less, it would follow that s' will be expanded before s. But, path to s might open doors much sooner than path to s'!

Pending further insights, it seems the state is forced to remember sequences of contrained nodes instead of merely sequences of constraints. This way, we guarantee prefix-suboptimality, and as a bonus get global ϵ-suboptimality without the extra factor of N.

Theorem 5. *When the algorithm terminates, it will return an $N\epsilon$-optimal plan, if it exists.*

Algorithm 1 $search()$

$jointCost = \infty$
while $N \max_{i=1}^{N} f_{min,i} < jointCost$ **do**
  choose $i \in \{1, \ldots, N\}$ such that $OPEN_i$ is not empty
  $(pos, hist_i)$:= remove from the front of $OPEN_i$
  if $pos = start_i$ **then**
    for all tuples $(hist_1, \ldots, hist_{i-1}, hist_{i+1}, \ldots, hist_N)$ **do**
      $jointCost := \min(jointCost, jointPriority($
        $(start_i, hist_i), \ldots, (start_N, hist_N)))$
    end for
  end if
  $expand(pos, hist)$
end while

Algorithm 2 $expand(pos, hist)$

mark $(pos, hist)$ as CLOSED
for all $(pos', hist') \in successors(pos, hist)$ **do**
  if $g(pos', hist') > g(pos, hist) + c(pos, pos')$ **then**
    $g(pos', hist') := g(pos, hist) + c(pos, pos')$
    $bp(pos', hist') := (pos, hist)$
    if $(pos', hist')$ is not CLOSED **then**
      insert $(pos', hist')$ in $agent.OPEN$
    end if
  end if
end for

Algorithm 3 $successors(u, seq_u)$

succlist := $\emptyset$
for all v such that $c(u, v) < \infty$ **do**
  $seq_v := seq_u$
  for all min constraints $\Phi^{\pm} \notin seq_u$ with $v \in \Phi^{\pm}$ **do**
    append $\Phi^{\pm}$ to seq_v
  end for
  for all max constraints $\Phi^{\pm}$ with $v \in \Phi^{\pm}$ **do**
    remove $\Phi^{\pm}$ from seq_v
    append $\Phi^{\pm}$ to seq_v
  end for
  push (v, seq_v) onto succlist
end for
return succlist

Algorithm 4 $greedyCost(\pi_1, \ldots, \pi_n)$

$\forall \Phi: \ trigger(\Phi) := \Phi$ is open or Φ^- is absent in $\pi_{1\ldots N}$
$\forall_1 \in 1 \ldots N: \ u_i := v_i := start_i$
$events := \{0 : \{1 \ldots N\}\}$
while $events$ is not empty **do**
  $(time, ids) := extractMin(events)$
  for all $i \in ids$ **do**
    $u_i := v_i$
    for all constraints Φ^- with $v \in \Phi^-$ **do**
      if Φ is an open constraint **or** Φ^- appears in no suffix of $\pi_{1\ldots N}$ after $u_{1\ldots N}$ **then**
        $trigger(\Phi) := true$
      end if
    end for
  end for
  for all $i \in 1 \ldots N$ with $u_i = v_i \neq goal_i$ **do**
    $v_i :=$ successor of u_i on π_i
    for all constraints Φ^+ with $v \in \Phi^+$ **do**
      if $\neg trigger(\Phi)$ **then**
        $v_i := u_i$ and continue with the next i
      end if
    end for
    $events(time + c(u_i, v_i)).insert(i)$
  end for
end while
if $u_{1\ldots N} = goal_{1\ldots N}$ **then**
  return $time$
else
  return ∞
end if

References

Bhattacharya, S.; Likhachev, M.; and Kumar, V. 2010. Multi-agent path planning with multiple tasks and distance

Algorithm 5 $jointPriority((pos_1, hist_1), \ldots, (pos_N, hist_N))$

$\forall i$, build π_i by following $bp()$ from $(pos_i, hist_i)$
for all $i \in 1 \ldots N$ such that $pos_i \neq start_i$ **do**
  extend π_i with a zero cost edge to a node that opens all doors the agent can open, followed by $start_i$ at cost $\epsilon h(s_i)$
end for
return $greedyCost(\pi_1, \ldots, \pi_n)$

constraints. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 953–959. IEEE.

Christofides, N. 1976. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, DTIC Document.

De Weerdt, M.; Ter Mors, A.; and Witteveen, C. 2005. Multi-agent planning: An introduction to planning and coordination. In *In: Handouts of the European Agent Summer*. Citeseer.

Ferner, C.; Wagner, G.; and Choset, H. 2013. Odrm* optimal multirobot path planning in low dimensional search spaces. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 3854–3859. IEEE.

Gabbay, D. M.; Hodkinson, I.; Reynolds, M.; and Finger, M. 1994. *Temporal logic: mathematical foundations and computational aspects*, volume 1. Clarendon Press Oxford.

Georgeff, M. 1988. Communication and interaction in multi-agent planning. *Readings in distributed artificial intelligence* 313:125–129.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on* 4(2):100–107.

Moshkovitz, D. 2012. The projection games conjecture and the np-hardness of ln n-approximating set-cover. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer. 276–287.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence* 219:40–66.

Van der Hoek, W., and Wooldridge, M. 2008. Multi-agent systems. *Foundations of Artificial Intelligence* 3:887–928.