Distributed Path Consensus Algorithm

Subhrajit Bhattacharya
Mechanical Engineering
and Applied Mechanics
University of Pennsylvania
Philadelphia, PA 19104
subhrabh@seas.upenn.edu

Maxim Likhachev
Computer and
Information Science
University of Pennsylvania
Philadelphia, PA 19104
maximl@seas.upenn.edu

Vijay Kumar
Mechanical Engineering
and Applied Mechanics
University of Pennsylvania
Philadelphia, PA 19104
kumar@me.upenn.edu

Abstract

Path planning methods based on discrete graph searches are highly effective in dealing with large-scale environments, arbitrarily shaped obstacles, frequent map updates and complex cost functions. When it comes to planning for teams of coupled agents however, typical graph search-based planning techniques quickly become infeasible due to the exponential growth of the dimensionality of the joint state-space. In this paper, we study the problem of path planning for teams of agents coupled with time-parameterized constraints on the distances from each other, the problem that often arises in planning for multi-agent robotic systems with limited communication range. We first pose this problem as searching N (the number of agents) graphs for cost-minimal paths that satisfy the distance constraints. We then show how the solution to this problem - path consensus - can be achieved in distributed fashion. Our theoretical analysis gives the conditions under which the algorithm converges to an optimal solution. Our experimental analysis shows that the algorithm converges fast for teams of up to six agents navigating cluttered environments.

Introduction

This paper studies the problem of path planning for teams of agents coupled with time-parameterized constraints on the distances between each other. A number of multi-agent planning problems fall into these category. For instance, robots navigating towards their respective goal locations while staying within the communication range is one of the common planning problems in multi-agent robotics. The time-parameterized distance constraints for this problem will be set as constant (time-invariant). Another typical application area is search and/or coverage in settings where robots must rendezvous periodically to exchange information and/or maintain relative distance constraints to enable communication. For example, the agents may be required to move to their respective goal locations while some of them need to meet and exchange map information every 30 seconds. In this case, the distance constraints in between the corresponding agents will be set to a small value at every 30 second timestep and will be set to infinity at all other timesteps.

We are interested in planning in a discretized environment in a decentralized fashion in order to enable scalability. Continuous motion planning is possible for such problems [5], but only practical in environments with moderate complexity. Thus, in this paper, we explore discrete path planning algorithms for multiple robots with constraints on intermediate positions.

In this paper we study how and when the one-shot joint state-space planning can be decomposed into planning with a series of lower-dimensional searches converging to an optimal solution. In particular, we first formulate the planning problem as searching N graphs for cost-minimal paths that satisfy the distance constraints. We will refer to this problem as *finding cost-minimal Path Consensus*. We then develop an algorithm, called *Distributed Path Consensus (DPC)* that repeatedly searches these graphs while gradually increasing the weight of the constraints until full convergence. The basic idea is to incorporate constraints through penalty functions in a fashion that is reminiscent of augmented Lagrangian techniques. As in augmented Lagrangian techniques [7], it is possible to show convergence to optimal solutions when the cost function and constraint functions are convex. Our theoretical analysis gives the conditions under which the algorithm is guaranteed to converge to an optimal solution. Our experimental analysis shows that the algorithm converges fast for teams of up to six agents operating in cluttered environments. In addition to the presented simulation results, we also show the implementation and execution of the algorithm on three physical robots navigating a cluttered environment while obeying time-parameterized distance constraints.

Related Works

Robot path planning is probably one of the most extensively studied problems areas in robotics [28]. Broadly speaking algorithms can be divided into two categories: continuous and discrete planning planning methods. In continuous planning methods, one tries to derive either open loop trajectories [47] or closed loop feedback policies [40, 12] that avoid obstacles while satisfying constraints on the robot dynamics. However, it is difficult to establish completeness and convergence results except in special cases. In most practical settings, discrete graph search methods have been shown to be complete and efficient [44]. And in some cases [29], it is even possible to incorporate robot dynamics to solve planning problems.

Multi-robot path planning suffers from the inherent complexity resulting from the necessity of operating in Cartesian products of configuration and state spaces [14]. The continuous path planning problem is even more difficult to solve in a centralized setting [47, 32] unless the problem is solved sequentially for each robot [48]. Open loop trajectory planning problems can be reduced to optimization problems. While completeness results are often possible [4] for simple or no constraints, it is difficult to respect more complex multi-robot constraints.

Closely related to our present work is the work on task allocation for multiple robots[22]. In these methods, one can impose rendezvous constraints at intermediate time points as tasks and reformulate the path planning problem as a task allocation problem. This then lends itself to auction-based solutions[16] for the team. However, these methods can produce grossly sub-optimal solutions in the environments with obstacles.

The iterative method proposed in the present work is closely related to those found in distributed optimization techniques [9] and to solving decentralized planning under uncertainty problems [6, 34, 42]. Such problem is often solved sequentially for each agent [48]. However, the novelty is our ability to find efficiently the optimal paths in arbitrarily complex environments with constraints on intermediate points.

Separable optimization problems [8] (optimization problems that can be split up into simpler sub-problems involving only certain partitions of the variable set) with linear constraints have been studied extensively in the past and solved in a distributed fashion using techniques based on dual decomposition [41, 8]. Augmented Lagrangian type methods have been used for solving similar problems more efficiently [7, 35]. However such methods are limited to problems with linear constraints and rely on convexity of cost functions.

Path planning methods based on discrete graph searches such as Dijkstra's, A* [17, 24] and its variants are a common way of planning for a single agent. These methods have been shown to be suitable for planning under time-constraints [27, 23, 30], are able to re-plan efficiently whenever terrain map is updated [43], and can accommodate various cost functions and agent dynamics and environment constraints [29]. However when planning for multi-agent systems the joint state-space of N agents requires the construction and search of a graph whose size grows exponentially as N increases. Real-time planning and re-planning in such large state-spaces becomes infeasible.

Our use of time-parameterized distance constraints is related to the work on task allocation for multiple agents [22]. In these methods, one can impose rendezvous constraints at intermediate time points as tasks and reformulate the path planning problem as a task allocation problem. This then lends itself to auction-based solutions [**?**] for the team. However, these methods can produce highly sub-optimal solutions. No completeness or optimality results are available for planning paths in the presence of obstacles with such methods.

There also exist an extensive research of distributed planning in a more general sense (a good survey can be found in [18]). For example, in Cooperative Distributed Planning (CDP) agents communicate and coordinate with an aim to reach a consensus on paths [15]. The DIPART experimental platform [38] can analyze a wide range of planning and execution problems, especially in relation to task allocation. Communication [25] and coordination [19, 20] are important aspects of such planning. The method we propose is also related to those found in distributed optimization techniques [9]. Our paper concentrates specifically on path planning under time-parameterized distance constraints and studies how and under what conditions, one can obtain cost-minimal path consensus without ever planning in the joint state-space.

Distributed optimization in iterative fashion by gradual increment of Lagrangian-like penalty weights has been used for solving shortest path problems with resource constraints [26]. The technique of Multiplier Adjustment Method (MAM) has been used in various integer and mixed integer optimization problems like set partitioning [11] and assignment problem [21]. However use of such a method for planning in a high dimensional space like that of robot trajectories is computationally highly expensive even with the proposed distributed scheme in [26]. This is because the the planning variable in our case are trajectories that lie in infinite dimensional Hilbert space. Even if we discetize the space, the number of planning variable required for defining a path will be extremely high. Moreover the resource constraint is not quite applicable to the problem of robot path planning.

In similar lines, Lagrangian dual problem has been used to solve shortest path problem in a network subject to knapsack-type or resource constraints [**?**]. However such approaches are highly specific to the resource-type constraints and is suitable for moderately sized network graphs. The graphs resulting from discretization of a continuous space, as in our case, is significantly larger compared to the network graphs and the distance-type constraints in our case are quadratic or even more complex than that.

Sven Koenig, et al. has investigated Distributed Constraint Optimization Problems (DCOPs). The BnB-ADOPT algorithm [46] does an A* search on the spanning tree of the constraint graph of the agents. The algorithm deals with assigning states to agents such that constraint costs are minimized. While this is once again more relevant to task allocation problem than path planning, it is possible to frame our present problem as a particular case of DCOP by describing the complete path for each agent as their states. Quite evidently, for using this algorithm in constrained path planning as in the present problem, if we describe each path for an agent as its state, the number of possible states for each agent will be extremely huge. In that case this algorithm will be very inefficient and computationally expensive to work with. The "Opportunistic Best-First Search", where the ADOPT algorithm performs sequential searches exchanging information between the agents until termination is attained, is in essence very similar to our present approach, but applicable to a different problem scenario.

Some of the other relevant works done in the past are elastic-based path planning [39] and potential-field based path planning [31]. In methods like these, distance constraints are not present as in our case, and hence the problem with high dimensionality of the joint state-space is not faced. Planning for multiple robots can be done independently - at least there is no guarantee provided by this methods for for multi-agent planning. Moreover, for the most generic environments these methods pose the problem of local energy/potential minima. Similar work of decentralized planning of multi-robot trajectories under collision and communication constraint using navigation functions have been investigated [37], but such approaches do not provide guarantee of optimality or completeness.

Probabilistic or sample-based path planning [45] on the other hand although can deal with constraints like the distance constraints, the typical search in such methods are performed through the 'super-graph' of the joint state-space of the agents (probabilistic roadmaps for example) and hence scales exponentially with the number of agents. Moreover being a sample-based method completeness is guaranteed only probabilistically.

Optimization of velocity profile for path constrained robots [3], maintainance of communication connectivity in an obstacle-free environment for robots with second-order dynamics [36] and maintainance of graph connectivity using potential functions [33] are some of the other related works worth mentioning, which although differ markedly from our present work in terms of the nature of the constraints, the generality in the kind of environment we consider and the kind of guarantee on completeness and optimality we can prove on our algorithm.

Problem Formulation, Assumptions, and Notations

In this section, we formalize the planning problem for multi-robot goal-directed navigation with distance constraints as the problem of finding a set of paths through N graphs that satisfy the constraints in between, where N is the number of robots. We will denote each robot by R_i, $1 \leq i \leq N$. We assume that the constraints functions are given by time parametrized maximum distance between each pair of robots, R_i and R_j. One particular case of such constraints is the rendezvous constraints, where the foresaid function takes small finite values at particular instants of time, and remain infinite at all other instants of time.

Goal-directed navigation Planning for goal-directed navigation for an individual robot R_i is often modeled as computing a least-cost path through a graph G_i formed by discretization of the configuration space. Each state $\mathbf{s} \in \mathbf{V}(G_i)$ is given by $\{x, y\}$ coordinates of the corresponding cell. For permissible and neighboring states $\mathbf{s}$ and $\mathbf{s}'$ (free states inside the configuration space) the edge $\mathbf{s} \rightarrow \mathbf{s}' \in \mathbf{E}(G_i)$ is associated with a strictly positive cost $c(\mathbf{s}, \mathbf{s}')$. A common choice for the costs is to be equal to the Euclidean distances (or its square) in between the centers of the corresponding cells.

Typically, for a single robot R_i, a planner would find a path through the graph G_i that connects states $Start_i$ and $Goal_i$ and minimizes the cumulative cost of transitions using a search algorithm. However, in our case, the distance constraints are specified as functions in time at which robots have to be within certain distance from each other. As a result, the generated paths need to be time-parameterized. To achieve this, we augment each state in the graph G_i with an additional variable - time index[1], such that an augmented state becomes $\{\mathbf{s}, t\}$. Edges in this augmented graph $H_i = G_i \times \{0, 1, \cdots, T\}$ are defined such that a particular node $\{\mathbf{s}, t\}$ connects only to the nodes $\{\mathbf{s}', t+1\}$, such that $\mathbf{s}' \in (Neighbors(\mathbf{s}) \cup \mathbf{s}) \subset \mathbf{V}(G_i)$. Planning in such an H_i ensures that the planning is done both in space and time, enabling incorporation of dynamic obstacles and inter-robot collision avoidance, and the time parametrized trajectory returned by the planner is consistent with the fact that the robots can move only forward in time.

For the simplicity of notations, we will assume that all paths of interest to us are at most T timesteps. Thus, a T-step path from $\{Start_i, 0\}$ to $\{Goal_i, T\}$ is given by the ordered set $\pi_i = \{s_0 = Start_i, s_1, \ldots, s_T = Goal_i\}$. Thus, $\pi_i(t)$ will refer to the state s_t the robot R_i is at time t when following path π_i. The cost of a path is given as $c(\pi_i) = \sum_{j=1\ldots T} c(s_{j-1}, \mathbf{s}_j)$.

We assume that the start and goal coordinates of each robot are given and fixed. Thus, when we write π_i to be a path in H_i, we immediately imply the constraint that the first node on the path is $\{Start_i, 0\}$ and the last node on the path is $\{Goal_i, T\}$. For notational simplicity we don't write this constraint explicitly every time. But whenever we write π_i we mean a trajectory for R_i such that $\pi_i(0) = Start_i$ and $\pi_i(T) = Goal_i$.

Constraints We represent constraints between pair of robots, R_i and R_j, as time parametrized functions of maximum distance between them. For any pair of states $\mathbf{s} \in \mathbf{V}(G_i)$, $\mathbf{s}' \in \mathbf{V}(G_j)$, where $i \neq j$, we define a distance $d(\mathbf{s}, \mathbf{s}')$ as a non-negative

[1] One should also be able to use non-uniform transition times. We assume constant times for each transition purely for the sake of simpler explanations.

finite scalar-valued function satisfying commutativity (e.g., $d(\mathbf{s}, \mathbf{s}') = d(\mathbf{s}', \mathbf{s})$). Thus, in case graphs G_i were derived from a 2D gridworld, the distance function $d(\mathbf{s}, \mathbf{s}')$ can be a simple Euclidean distance in between the centers of the cells that correspond to state $\mathbf{s}$ for the first robot and state $\mathbf{s}'$ for the second robot. In other cases, the distance function can model more complex factors.

We thus specify time-parameterized distance constraints between all pairs of the robots $\phi_{i,j}$ for all $i \neq j$. Thus, $\phi_{i,j}$ is a vector of T scalar values such that $\phi_{i,j}(t)$ implies that the distance $d(\cdot, \cdot)$ in between robots R_i and R_j at time t should be no more than $\phi_{i,j}(t)$. The $\phi_{i,j}(t) = \infty$ therefore implies the absence of any constraint in between these robots at time t.

Objective function Given our formulation of the problem, the goal of an optimal planning algorithm would be to find N paths π_i^* $(1 \leq i \leq N)$ through the corresponding graphs G_i such that:

$$\{\pi_1^*, \ldots, \pi_N^*\} = \text{argmin}_{\pi_1 \ldots \pi_N} \textstyle\sum_{j=1 \ldots N} c(\pi_j) \tag{1}$$

subject to the constraint that

$$\begin{array}{l} d(\pi_i(t), \pi_j(t)) \leq \phi_{i,j}(t) \\ \text{for every } \{i, j, t\} \text{ s.t. } 1 \leq t \leq T, 1 \leq i \leq N, 1 \leq j \leq N, i \neq j \end{array} \tag{2}$$

DPC Algorithm

We explain the DPC algorithm in two steps. We first present the version of the algorithm suitable for solving the multi-robot goal-directed navigation problems with distance constraints in spaces free of obstacles. In the next section, we therefore explain an extended version of the DPC algorithm that was specifically developed for fast convergence in the presence of obstacles. It runs the basic version of the algorithm repeatedly (in superiterations), thus making it more computationally expensive, but at the same time guarantees completeness.

DPC Algorithm without Superiterations

The basic idea behind the DPC algorithm without superiterations is to run a series of graph searches on graphs H_i, $1 \leq i \leq N$. At each iteration $iter$, the search processes some graph H_r and computes a path that minimizes the weighted sum of the pathcost plus the amount to which the paths violate the rendezvous point constraints with respect to the paths computed for other robots previously. In that respect, the algorithm is similar to [34]. The difference is that DPC slowly increases the weights associated with the rendezvous point constraints. This makes the robots to increase their pathcosts slowly and often converge to an optimal solution. We also present the proof of convergence, completeness and optimality of the algorithm in an empty environment and under some specific assumptions.

```
1  procedure BasicDPC()
2  compute $\pi_i^0 = \text{argmin}_{\pi_i} c(\pi_i)$ for all $i$;
3  set $w_{i,j}^0 = 0$ for all $i, j$;
4  $r = 1, iter = 0$;
5  while ($\sum_{i=1\ldots N} \sum_{j=i+1\ldots N} \Omega(\pi_i^{iter}, \pi_j^{iter}) \neq 0$ AND
         $\sum_i c(\pi_i^{iter}) + \sum_{\substack{i,j\\ i\neq j}} w_{i,j}^{iter} \Omega(\pi_i^{iter}, \pi_j^{iter}) \leq N * maxpathcost$)
6      set $w_{i,j}^{iter+1} = w_{i,j}^{iter}$ for all $i, j$;
7      $w_{i,j}^{iter+1} = w_{j,i}^{iter+1} = w_{j,i}^{iter} + \epsilon_{i,j}^k$ for all $i, j$;
8      compute $\pi_r^{iter+1} = \text{argmin}_{\pi_r} \{c(\pi_r) + \sum_{i=1\ldots N, i\neq r} w_{i,r}^{iter+1} \Omega(\pi_i^{iter}, \pi_r)$
                 $+ \sum_{j=1\ldots N, j\neq r} w_{r,j}^{iter+1} \Omega(\pi_r, \pi_j^{iter})\}$;
             $= \text{argmin}_{\pi_r} \{c(\pi_r) + 2\sum_{i=1\ldots N, i\neq r} w_{i,r}^{iter+1} \Omega(\pi_i^{iter}, \pi_r)\}$;
9      set $\pi_j^{iter+1} = \pi_j^{iter}$ for all other $j \neq r$;
10     $iter = iter + 1$;
11     $r = r + 1$;
12     if $r > N$
13         $r = 1$;
```

Figure 1: DPC without superiterations

To penalize for the violation of constraints, the algorithm introduces the penalty function function $\Omega(\pi_i, \pi_j)$, $i \neq j$ as follows:

$$\begin{aligned} &\Omega(\pi_i, \pi_j) = \textstyle\sum_{t=0,1,\cdots,T} \varpi(\pi_i(t), \pi_j(t), \phi_{i,j}(t)) \\ &\text{where, } \varpi(\mathbf{s}, \mathbf{s}', p) = \max(0, d(\mathbf{s}, \mathbf{s}') - p) \end{aligned} \quad (3)$$

Note that Ω depends on $\phi_{i,j}$ as well. However since typically $\phi_{i,j}$ is a constant throughout the problem (i.e. does not change with the iterations), for notational simplicity we use $\Omega(\pi_i, \pi_j, \phi_{i,j}) \equiv \Omega_{ij}(\pi_i, \pi_j) \equiv \Omega(\pi_i, \pi_j)$. The convention is that we look at the indices of the π's to identify which ϕ to use for defining Ω.

The penalty function is a way of expressing the hard distance constraints into soft constraints. To gradually increase the weight of each constraint violation, each constraint $\phi_{i,j}$ is associated with a dynamically adjusted weight w_{ij} (the weights have to be symmetric, so $w_{ij} = w_{ji}$).

The pseudocode of the algorithm is shown in figure 1. It first computes unconstrained least-cost paths for each of the robot (line 2). The computation of the paths can be done with any graph search such as A* search [24]. After that, the algorithm iterates over the robots repeatedly (robot index is incremented on lines 11-13). Within each iteration $iter$, the algorithm increases the penalty weights by a small increment ϵ

(line 7), which has the following property,

$$\epsilon_{i,j}^{k} \begin{cases} \geq 0, & \forall i = r \text{ or } j = r, \;\; i \neq j \\ = 0, & \text{otherwise} \end{cases} \tag{4}$$

It then computes a path for the robot R_r from its start state $\{Start_r, 0\}$ to its goal state $\{Goal_r, T\}$ in graph H_r that minimizes the summation of all the transition costs and all the weighted penalties (line 8). To compute this path, one can once again use any graph search for a least-cost path, but the costs of all the edges, however, need to be modified. In particular, the cost of any transition $\{\mathbf{s}, t-1\} \rightarrow \{\mathbf{s}', t\}$ becomes:

$$c(s, s') + 2 \sum_{j=1\ldots N, j\neq r} w_{rj}^{iter+1} \varpi(s', \pi_j^{iter}(t), \phi_{r,j}(t))$$

Example 1. *Figure 2 demonstrates frame-by-frame how the iterations of the DPC Algorithm without superiterations converge to a feasible solution by changing the trajectories of each robot change in small steps at every iteration. The simulation consists of 6 robots in an empty environment of size 1 unit by 1 unit, discretized into 100 by 100 cells. All the robots start at time $t = 0$ units and need to reach their goals at the time that is no later than $t = 8$ units. Time is discretized at the resolution of 0.1 units. The constraint between robot i and robot $i+1$, for $i = 1, 3, 5$ is that the distance between them needs to be 0.0 units at time $t = 2.0$ & $t = 6.0$, and for $i = 2, 4$ the distance between the robots needs to be 0.0 units at time $t = 4.0$. At any other time, there is no constraint on the distances in between the robots.*

The first iteration is the unconstrained planning which returns the straight line trajectories. In iteration 2, Robot 1 plans with the soft constraint of rendezvousing with Robot 2 at two points on its trajectory. Thus, the trajectory of robot 1 changes accordingly. In iteration 3, Robot 2 plans its trajectory based on the most recent trajectories of all the robots. Its constraints are to meet Robot 1 at two points and robot 3 at a single point in the middle of its trajectory. The changes in its trajectory are clearly directed towards satisfying these constraints. In this fashion, the robots sequentially make small changes to their trajectories gradually progressing towards the global feasible solution. The solution converges in 13 iterations. Each iteration is a single A search in a 3-dimensional space (H_i), as opposed to a full $12+1$-dimensional joint state-space of 6 robots. The number of states in each H_i is 800,000. The number of iterations required for the convergence is 13, and the time taken for the overall planning was less than 10 seconds. Figure 2(m) shows the final converged solution. Also, it can be noted that since it is an 8-connected grid, the motions of the robots are restricted in their orientations, and the generated solution is just one of the many optimal solutions as was discussed in section .*

The changes in its trajectory with the increase in the penalty weights are clearly directed towards satisfying the constraints. The robots sequentially make small changes to their trajectories gradually progressing towards the global feasible solution.

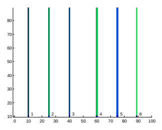
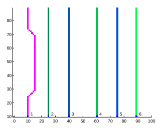
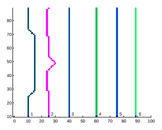
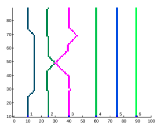
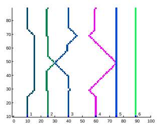
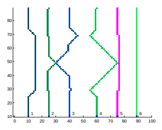
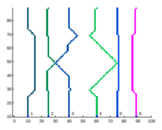
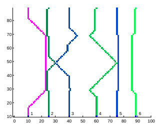
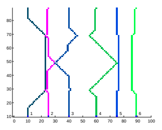
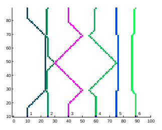
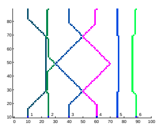
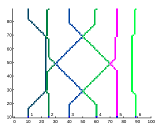
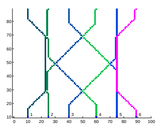

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

(k)

(l)

(m)

Figure 2: Planning in an empty environment (Example 1) with 6 robots using $BasicDPC$ without superiterations. Each frame represent an iteration and the planning robot R_r's trajectory is colored in pink.

Theoretical Properties for two agents

In the following sections π_i will indicate a trajectory of robot i consistent with its initial and goal configurations at $Start_i$ and $Goal_i$ respectively, even if that is not explicitly mentioned. This means π_i belongs to a subspace of the space of all possible trajectories sich that the initial and goal configurations at $Start_i$ and $Goal_i$.

For the two-robot case, since the weighs and the increments need to be symmetric, for notational simplicity we define $w \equiv 2w_{1,2} = 2w_{2,1}$ and $\epsilon \equiv 2\epsilon_{1,2} = 2\epsilon_{2,1}$. Consider the case when two agents are planning in an environment without obstacles. Let's define the following for this scenario (recall that $\Omega(\pi_a, \pi_b) = \Omega(\pi_b, \pi_a)$):

$$\begin{aligned}\Psi(w, \pi_1) &= \min_{\pi_2'}(c(\pi_2') + w \cdot \Omega(\pi_1, \pi_2')) \\ \Psi(w, \pi_2) &= \min_{\pi_1'}(c(\pi_1') + w \cdot \Omega(\pi_1', \pi_2))\end{aligned}$$

Definition 1 (well-behaved cost and distance functions). *The function pair $\{c, d\}$ is said to be well-behaved iff there exists small enough $\epsilon_0 > 0$ such that for any $\epsilon_0 \geq \epsilon > 0$ it holds that*

$$\Psi(w+\epsilon, \pi_1^{*w}) - \Psi(w, \pi_1^{*w}) \leq \Psi(w+\epsilon, \pi_1^{*w+\epsilon}) - \Psi(w, \pi_1^{*w+\epsilon})$$
$$\Psi(w+\epsilon, \pi_2^{*w}) - \Psi(w, \pi_2^{*w}) \leq \Psi(w+\epsilon, \pi_2^{*w+\epsilon}) - \Psi(w, \pi_2^{*w+\epsilon})$$

for any $\infty > w \geq 0$ and any paths π_1 or π_2, where for a given w we define

$$\begin{aligned}\{\pi_1^{*w}, \pi_2^{*w}\} &= argmin_{\pi_1', \pi_2'}\{c(\pi_1') + c(\pi_2') + \\ &\quad w \cdot \Omega(\pi_1', \pi_2')\}\end{aligned}$$

Note that this definition is also dependent on the structure and discretization of the Graph G_i since the function Ψ takes in π as an input, which in turn depends on the discretization of the graph.

In this condition, the pair $\{\pi_1^{*w}, \pi_2^{*w}\}$ represents a cost-minimal pair given the cost function formulated as the sum of the path costs and the weighted penalty function, and the pair $\{\pi_1^{*w+\epsilon}, \pi_2^{*w+\epsilon}\}$ is the same but for the weight equal to $w + \epsilon$. Therefore, in plain English, the condition above requires the increase in Ψ function due to the increase in w (by ϵ) to be smaller whenever the current path pair is cost-minimal. If this condition is satisfied, then the algorithm exhibits the following cornerstone property: after every iteration the current set of paths is cost-minimal w.r.t the cost function formulated as the sum of the path costs and the weighted penalty function.

Theorem 1. *Given the condition defined in Definition 1 holds and a sufficiently small ϵ is used in the algorithm, every time line 5 is executed the following holds:*

$$\{\pi_1^{iter}, \pi_2^{iter}\} = \qquad \arg\min_{\pi_1, \pi_2}\left(\sum_{i=1\ldots N} c(\pi_i) \ + \ w^{iter} \cdot \Omega(\pi_1, \pi_2)\right)$$

Proof. The theorem clearly holds when line 5 is executed for the first time since $w_{i,j}^0 = 0$ for all i, j.

We now prove the theorem by induction. Assume it holds for 0th through $iter = 0 \ldots k$ executions of line 5. We will now prove that it continues to hold for the $iter = k + 1$th execution of line 5. By inductive assumption we have:

$$\{\pi_1^k, \pi_2^k\} = \arg\min_{\pi_1, \pi_2} c(\pi_1) + c(\pi_2) + w^k \cdot \Omega(\pi_1, \pi_2)$$

Without loss of generality, let's assume that at $iter = k^{th}$ execution of line 5, $r = 1$. In other words, at k^{th} iteration, the algorithm was computing the path for the 1st robot, π_1^{k+1}, and therefore $\pi_2^{k+1} = \pi_2^k$.
We prove by contradiction. Let us assume that

$$\{\pi_1^{k+1}, \pi_2^{k+1}\} \neq \arg\min_{\pi_1, \pi_2}(c(\pi_1) + c(\pi_2) + w^{k+1} \cdot \Omega(\pi_1, \pi_2))$$

This implies that there exist π_1' and π_2' such that
$\{\pi_1', \pi_2'\} = \arg\min_{\pi_1, \pi_2}(c(\pi_1) + c(\pi_2) + w^{k+1} \cdot \Omega(\pi_1, \pi_2))$

and

$$c(\pi_2^{k+1}) + c(\pi_1^{k+1}) + w^{k+1} \cdot \Omega(\pi_1^{k+1}, \pi_2^{k+1}) \\ > c(\pi_2') + c(\pi_1') + w^{k+1} \cdot \Omega(\pi_1', \pi_2') \tag{5}$$

Since the algorithm has just computed π_1^{k+1} it holds that:
$\pi_1^{k+1} = \arg\min_{\pi_1} c(\pi_1) + w^{k+1} \cdot \Omega(\pi_1, \pi_2^{k+1})$
Also, it holds that $\pi_1' = \arg\min_{\pi_1}(c(\pi_1) + w^{k+1} \cdot \Omega(\pi_1, \pi_2'))$ and $\pi_2' = \arg\min_{\pi_2}(c(\pi_2) + w^{k+1} \cdot \Omega(\pi_1', \pi_2))$. Thus from 5,

$$c(\pi_2^{k+1}) + \min_{\pi_1}(c(\pi_1) + w^{k+1} \cdot \Omega(\pi_1, \pi_2^{k+1})) \\ > c(\pi_2') + \min_{\pi_1}(c(\pi_1) + w^{k+1} \cdot \Omega(\pi_1, \pi_2')) \tag{6}$$

$$\Rightarrow \quad c(\pi_2^{k+1}) - c(\pi_2') > \Psi(w^{k+1}, \pi_2') - \Psi(w^{k+1}, \pi_2^k) \tag{7}$$

On the other hand, since $\{\pi_1^k, \pi_2^k\}$ is a global minimum for w^k according to our inductive assumption, and since by hypothesis $\pi_2^k = \pi_2^{k+1}$, it holds that

$$c(\pi_2^k) + \Psi(w^k, \pi_2^k) \quad \leq \quad c(\pi_2') + \Psi(w^k, \pi_2') \tag{8}$$

$$c(\pi_2^{k+1}) - c(\pi_2') \quad \leq \quad \Psi(w^k, \pi_2') - \Psi(w^k, \pi_2^k) \tag{9}$$

Thus, from inequalities 6 and 8, and using $\pi_2^k = \pi_2^{k+1}$,

$$\Psi(w^k, \pi_2') - \Psi(w^k, \pi_2^{k+1}) > \Psi(w^{k+1}, \pi_2') - \Psi(w^{k+1}, \pi_2^{k+1})$$

$$\Rightarrow \quad \Psi(w^k + \epsilon, \pi_2') - \Psi(w^k, \pi_2') < \Psi(w^k + \epsilon, \pi_2^k) - \Psi(w^k, \pi_2^k)$$

But the pair of paths $\{\pi_1^k, \pi_2^k\}$ are the global minimum for $w = w^k$ by the inductive assumption. Moreover, $\pi_2' = \pi_2^{*w+\epsilon}$ by hypothesis. From Definition 1 it therefore follows that:

$$\Psi(w^k + \epsilon, \pi_2^k) - \Psi(w^k, \pi_2^k) \\ \leq \Psi(w^k + \epsilon, \pi_2') - \Psi(w^k, \pi_2')$$

Thus from the last two inequalities we end up with a contradiction. Hence our assumption of the existence of $\{\pi_1', \pi_2'\}$ was incorrect.

□

Since theorem 1 basically says that after every iteration the solution paths are cost-minimal w.r.t. the current weight w, the optimality of the algorithm follows from the fact that for a sufficiently large weight w, the weighted penalty function will dominate completely the cost function $c(\cdot)$, and therefore the paths will converge to the cost-minimal path consensus if there exists one. It happens in finite amount of time, since weights are increased by a strictly positive (bounded from below) increment ϵ and assuming there is a bound on the cost of a shortest path (denoted by $maxpathcost$). The following theorem states this formally.

Theorem 2. *Given the condition defined in 1 holds and a sufficiently small ϵ is used in the algorithm, the algorithm terminates, and when it terminates (say at iteration $iter$) the following holds:*

- $\{\pi_1^{iter}, \pi_2^{iter}\} = \{\pi_1^*, \pi_2^*\}$ *if there exists any set of paths that minimizes the objective function 1 and satisfies the constraints 2;*
- $\{\pi_1^{iter}, \pi_2^{iter}\} = \arg\min_{\pi_1,\pi_2} \big(c(\pi_1) + c(\pi_2) + w^{iter} \cdot \Omega(\pi_1, \pi_2)\big)$ *otherwise;*

Proof. As before, without loss of generality, we assume that at $iter = k^{th}$ execution of line 5, $r = 1$. From result of Theorem 1, and using the fact that $c(\pi_2^k) = c(\pi_2^{k+1})$, we have,

$$\begin{aligned} & c(\pi_1^k) + c(\pi_2^k) + w^k \cdot \Omega(\pi_1^k, \pi_2^k) \\ & \qquad \le c(\pi_1^{k+1}) + c(\pi_2^{k+1}) + w^k \cdot \Omega(\pi_1^{k+1}, \pi_2^{k+1}) \\ \Rightarrow \quad & w^k \cdot \Omega(\pi_1^k, \pi_2^k) - w^k \cdot \Omega(\pi_1^{k+1}, \pi_2^{k+1}) \le c(\pi_1^{k+1}) - c(\pi_1^k) \end{aligned}$$

Similarly,

$$c(\pi_1^{k+1}) - c(\pi_1^k) \quad \le \quad w^{k+1} \cdot \Omega(\pi_1^k, \pi_2^k) - w^{k+1} \cdot \Omega(\pi_1^{k+1}, \pi_2^{k+1})$$

Thus, combining the above two and rearranging,

$$\begin{aligned} & w^k \cdot (\Omega(\pi_1^k, \pi_2^k) - \Omega(\pi_1^{k+1}, \pi_2^{k+1})) \\ & \qquad \le w^{k+1} \cdot (\Omega(\pi_1^k, \pi_2^k) - \Omega(\pi_1^{k+1}, \pi_2^{k+1})) \end{aligned}$$

Since $0 \le w^k < w^{k+1}$, therefore we should have,

$$\begin{aligned} & \Omega(\pi_1^k, \pi_2^k) - \Omega(\pi_1^{k+1}, \pi_2^{k+1}) \ge 0 \\ \Rightarrow \quad & \Omega(\pi_1^{k+1}, \pi_2^{k+1}) \le \Omega(\pi_1^k, \pi_2^k) \end{aligned}$$

Thus we see that in every iteration step the violation cost decreases. The algorithm terminates to a feasible solution only when the violation cost becomes zero (step 5 of the algorithm). Then the solution to the problem becomes, by Theorem 1,

$$\begin{aligned} & \{\pi_1^{iter}, \pi_2^{iter}\} = \\ & \qquad \arg\min_{\pi_1,\pi_2} \textstyle\sum_{i=1}^{2} c(\pi_i) + w^{iter} \cdot 0 \\ \Rightarrow \quad & \{\pi_1^{iter}, \pi_2^{iter}\} = \{\pi_1^*, \pi_2^*\} \end{aligned}$$

Otherwise, termination to an infeasible solution will trivially imply

$$\{\pi_1^{iter}, \pi_2^{iter}\} = \arg\min_{\pi_1,\pi_2} \Big(c(\pi_1) + c(\pi_2) + w^{iter} \cdot \Omega(\pi_1, \pi_2)\Big)$$

Now we justify the proposed termination criteria when $\Omega(\pi_1^k, \pi_2^k) \neq 0$ at termination. From result of Theorem 1, and using the fact that $w^k < w^{k+1} = w^k + \epsilon$

$$\begin{aligned}
c(\pi_1^k) &+ c(\pi_2^k) + w^k \cdot \Omega(\pi_1^k, \pi_2^k) \\
&\leq c(\pi_1^{k+1}) + c(\pi_2^{k+1}) + w^k \cdot \Omega(\pi_1^{k+1}, \pi_2^{k+1}) \\
&< c(\pi_1^{k+1}) + c(\pi_2^{k+1}) + w^{k+1} \cdot \Omega(\pi_1^{k+1}, \pi_2^{k+1})
\end{aligned}$$

Since we have a discretized space, we can have a measurably finite Δ such that

$$\Delta \quad = \min_{\pi_1,\pi_2}\{\Omega(\pi_1, \pi_2) \text{ s.t. } \Omega(\pi_1, \pi_2) \neq 0\}$$

Then,

$$\begin{aligned}
c(\pi_1^k) &+ c(\pi_2^k) + w^k \cdot \Omega(\pi_1^k, \pi_2^k) \\
&\leq c(\pi_1^{k+1}) + c(\pi_2^{k+1}) + w^{k+1} \cdot \Omega(\pi_1^{k+1}, \pi_2^{k+1}) - \Delta\epsilon
\end{aligned}$$

Thus the total cost grows at least linearly with k unless $\Omega(\pi_1^k, \pi_2^k) = 0$. This will guarantee that the infeasibility termination criteria is achieved unless of course a feasible solution is found before that. That is, the value of $w_{1,2}^k \cdot \Omega(\pi_1^k, \pi_2^k)$ does not asymptotically tend to a positive value as k tends to infinity. [Note: Although the termination criteraia is on the violation cost rather than the total cost, we can note that the terms $c(\pi_1^k) + c(\pi_2^k)$ are bounded by $2 * maxpathcost$. This will imply that only the violation cost portion is able to increase linearly or faster with k.])

Next we will prove that if the infeasibility criteria is indeed achieved, then there does not exist a feasible solution.

Termination due to the second condition (infeasibility condition) implies (at the iteration k where the termination takes place),

$$c(\pi_1^k) + c(\pi_2^k) + w^k \cdot \Omega(\pi_1^k, \pi_2^k) > 2 * maxpathcost$$

and, by Theorem 1,

$$\{\pi_1^k, \pi_2^k\} = argmin_{\pi_1,\pi_2} c(\pi_1) + c(\pi_2) + w^k \cdot \Omega(\pi_1, \pi_2)$$

By contradiction, let us assume that there exists a feasible solution $\{\pi_1^{'*}, \pi_2^{'*}\}$ so that,

$$\begin{aligned}
\{\pi_1^{'*}, \pi_2^{'*}\} \quad &= \quad argmin_{\pi_1,\pi_2} c(\pi_1) + c(\pi_2) \\
&\qquad \text{s.t. } \Omega(\pi_1, \pi_2) = 0
\end{aligned}$$

If $\{\pi_1^k, \pi_2^k\} \neq \{\pi_1^{'*}, \pi_2^{'*}\}$, then by Theorem 1 we should have,

$$\begin{aligned}
& c(\pi_1^k) + c(\pi_2^k) + w^k \cdot \Omega(\pi_1^k, \pi_2^k) \\
&\qquad \leq c(\pi_1^{'*}) + c(\pi_2^{'*}) + w^k \cdot \Omega(\pi_1^{'*}, \pi_2^{'*}) \\
\Rightarrow \quad & c(\pi_1^k) + c(\pi_2^k) + w^k \cdot \Omega(\pi_1^k, \pi_2^k) \\
&\qquad \leq c(\pi_1^{'*}) + c(\pi_2^{'*}) \\
&\qquad\qquad [\because \Omega(\pi_1^{'*}, \pi_2^{'*}) = 0]
\end{aligned} \tag{10}$$

Thus, by using the termination criteria we have,

$$c(\pi_1^*) + c(\pi_2^*) \quad > \quad 2 * maxpathcost$$

Which is not possible in a discrete graph of finite size. Thus there can't exist $\{\pi_1^*, \pi_2^*\}$ as hypothesized.

□

Satisfying the condition of well-behaved function defined in Definition 1

We now analyze when the condition defined in Definition 1 is satisfied assuming that G_i is continuous (i.e. $G_i \subset \mathbb{R}^2$). To do this, let us first introduce several additional notations.

Definition 2 (Best Trajectory functions)**.** *We define the following functions for a given w and call these 'best trajectory functions' (they are similar to the concept of best strategy in Game theoretic sense).:*

$$\Pi_1^{*w}(\pi_2) \quad = \quad argmin_{\pi_1'}\{c(\pi_1') + w \cdot \Omega(\pi_1', \pi_2)\} \tag{11}$$

$$\Pi_2^{*w}(\pi_1) \quad = \quad argmin_{\pi_2'}\{c(\pi_2') + w \cdot \Omega(\pi_2', \pi_1)\} \tag{12}$$

Definition 3 (Notations for derevatives)**.** *In general the trajectories, π, may be elements of an Hilbert space, $\pi : \mathbb{R} \rightarrow \mathbb{R}^2$ such that $\pi(t)$ gives the coordinate of the trajectory at time t. However for simplicity we can assume that the time is discretized, and hence π is just a vector whose elements are the x and y coordinates of the trajectory points at the discretized time instants. c being a scalar function of the trajectories, it's derivative w.r.t. the trajectory has the simple interpretation of the gradient vector. The second derivative will in similar way be interpreted as a Hessian matrix, and the higher derivatives are higher order tensors. Similar interpretations can be made for the derivatives of Ω.*

We use the following notations for simplicity:

$$c^{(k)} \quad \equiv \quad \frac{\partial^k c}{\partial \pi^k}$$

$$\Omega^{(k,l)} \quad \equiv \quad \frac{\partial^{(k+l)}\Omega}{\partial {\pi_1}^k \partial {\pi_2}^l}$$

where, π_1 and π_2 denote respectively the first and second arguments of Ω.

This implies $\pi_1^{*w} = \Pi_1^{*w}(\pi_2^{*w})$ and $\pi_2^{*w} = \Pi_2^{*w}(\pi_1^{*w})$ (This is analogous to a Nash Equilibrium for this game).

Hypothesis 1 (Hypotheses on c and Ω)**.** *Consider the following hypotheses. We will show that the conditions for well-behaved function in Definition 1 hold if these hypotheses are true, and later we will discuss under what circumstances these hypotheses will hold.*

The hypotheses are as follows:

i. $\Omega^{(1,0)}(\pi_a, \pi_b) = -\Omega^{(0,1)}(\pi_a, \pi_b) \qquad \forall \pi_a, \pi_b$

ii. $\Omega^{(2,0)}(\pi_a, \pi_b) = \Omega^{(0,2)}(\pi_a, \pi_b) \succ 0 \qquad \forall \pi_a, \pi_b$

iii. $\Omega^{(1,1)}(\pi_a, \pi_b) \preceq 0 \qquad \forall \pi_a, \pi_b$

iv. $c^{(2)}(\pi) = K(const.) \succ 0 \qquad \forall \pi$

where the curly inequalities indicate positive or negative definiteness of the matrices.

We also note that,

$$\begin{aligned} & \Psi(w, \pi_2) = \min_{\pi_1'}(c(\pi_1') + w \cdot \Omega(\pi_1', \pi_2)) \\ \Rightarrow \quad & \Psi(w, \pi_2) = c(\Pi_1^{*w}(\pi_2)) + w \cdot \Omega(\Pi_1^{*w}(\pi_2), \pi_2), \quad \text{and} \\ & c^{(1)}(\Pi_1^{*w}(\pi_2)) + w \cdot \Omega^{(1,0)}(\Pi_1^{*w}(\pi_2), \pi_2) = 0 \end{aligned}$$

Thus,

$$\begin{aligned} \frac{\partial \Psi}{\partial w}(w, \pi_2) &= \left(c^{(1)}(\Pi_1^{*w}(\pi_2)) + w \cdot \Omega^{(1,0)}(\Pi_1^{*w}(\pi_2), \pi_2)\right) \frac{\partial \Pi_1^{*w}}{\partial w}(\pi_2) + \Omega(\Pi_1^{*w}(\pi_2), \pi_2) \\ &= \Omega(\Pi_1^{*w}(\pi_2), \pi_2) \qquad (13) \end{aligned}$$

Similarly,

$$\frac{\partial \Psi}{\partial w}(w, \pi_1) = \Omega(\pi_1, \Pi_2^{*w}(\pi_1)) \qquad (14)$$

Lemma 1 (Equivalent forms of well-behaved condition)**.** *As $\epsilon \to 0$, from the 'well-behaved function condition' we get for R_1,*

$$\begin{aligned} \Psi(w + \epsilon, \pi_1^{*w}) - \Psi(w, \pi_1^{*w}) &\leq \Psi(w + \epsilon, \pi_1) - \Psi(w, \pi_1) \\ \iff \frac{\partial \Psi}{\partial w}(w, \pi_1^{*w}) &\leq \frac{\partial \Psi}{\partial w}(w, \pi_1) \end{aligned}$$

for any $\pi_1 \neq \pi_1^{*w}$.
Similarly, the condition for trajectories of R_2:

$$\frac{\partial \Psi}{\partial w}(w, \pi_2^{*w}) \leq \frac{\partial \Psi}{\partial w}(w, \pi_2)$$

Using 13 and 14 this implies,

$$\begin{aligned} \Omega(\pi_1^{*w}, \pi_2^{*w}) &\leq \Omega(\pi_1, \Pi_2^{*w}(\pi_1)) \qquad \forall \pi_1 \qquad (15) \\ \Omega(\pi_1^{*w}, \pi_2^{*w}) &\leq \Omega(\Pi_1^{*w}(\pi_2), \pi_2) \qquad \forall \pi_2 \qquad (16) \end{aligned}$$

Or equivalently, using the condition of vanishing derevative at an extremum, the above two becomes,

$$\begin{aligned} \Omega^{(1,0)}(\pi_1^{*w}, \pi_2^{*w}) + T\, \Omega^{(0,1)}(\pi_1^{*w}, \pi_2^{*w}) = 0 \qquad (17) \\ \Omega^{(0,1)}(\pi_1^{*w}, \pi_2^{*w}) + T^{-1}\, \Omega^{(1,0)}(\pi_1^{*w}, \pi_2^{*w}) = 0 \qquad (18) \end{aligned}$$

where, $T = \left[\frac{\partial \Pi_2^{*w}}{\partial \pi_1} \Big|_{\pi_1^{*w}, \pi_2^{*w}} \right]$ and $T^{-1} = \left[\frac{\partial \Pi_1^{*w}}{\partial \pi_2} \Big|_{\pi_1^{*w}, \pi_2^{*w}} \right]$ can be interpreted as jacobians.

Finally on adopting the hypotheses 1.i., the above two equations reduce to the following final equivalent form of the 'well-behaved' condition,

$$\begin{aligned} & [I-T]\,\Omega^{(1,0)}(\pi_1^{*w},\pi_2^{*w}) = 0 \\ \Rightarrow \quad & T = I \end{aligned}$$

where, I is the identity element from the space where T resides. Thus the problem of justifying the 'well-behaved' condition now reduces to proving that $T = I$.

By definition, from 11 we have,

$$c^{(1)}(\Pi_1^{*w}(\pi_2)) + w\cdot\Omega^{(1,0)}(\Pi_1^{*w}(\pi_2),\pi_2) = 0 \tag{19}$$

$$c^{(1)}(\Pi_2^{*w}(\pi_1)) + w\cdot\Omega^{(0,1)}(\pi_1,\Pi_2^{*w}(\pi_1)) = 0 \tag{20}$$

Now, taking derevative of 19 w.r.t. π_2 and evaluating it at π_2^{*w} we get,

$$T^{-1}c^{(2)}(\pi_1^{*w}) + T^{-1}w\cdot\Omega^{(2,0)}(\pi_1^{*w},\pi_2^{*w}) + w\cdot\Omega^{(1,1)}(\pi_1^{*w},\pi_2^{*w}) = 0 \tag{21}$$

Ans similarly,

$$Tc^{(2)}(\pi_2^{*w}) + Tw\cdot\Omega^{(0,2)}(\pi_1^{*w},\pi_2^{*w}) + w\cdot\Omega^{(1,1)}(\pi_1^{*w},\pi_2^{*w}) = 0 \tag{22}$$

Substracting 21 from 22,

$$\begin{aligned} & Tc^{(2)}(\pi_2^{*w}) + Tw\cdot\Omega^{(0,2)}(\pi_1^{*w},\pi_2^{*w}) \\ & \quad -T^{-1}c^{(2)}(\pi_1^{*w}) - T^{-1}w\cdot\Omega^{(2,0)}(\pi_1^{*w},\pi_2^{*w}) = 0 \end{aligned} \tag{23}$$

But from 23 and using hypotheses 1.*ii.* and 1.*iv.* we get

$$\begin{aligned} & \left(K + \Omega^{(0,2)}(\pi_1^{*w},\pi_2^{*w})\right)(T - T^{-1}) = 0 \\ \Rightarrow \quad & T = \pm I \end{aligned}$$

The solution $T = -I$ is eliminated because in that the left hand side of 21 or 22 becomes negative definite violating the equalities. Hence we have $T = I$, which proves the equivalent form of the convexity condition in Lemma 1.

Now we will investigate the proposed hypotheses and check whether it holds true for the functional forms of c and Ω that were proposed.

First we check the function $\Omega(\pi_1,\pi_2) = \|\pi_1 - \pi_2\|^2$. Clearly, $\Omega^{(1,0)}(\pi_1,\pi_2) = 2(\pi_1 - \pi_2)$ and $\Omega^{(0,1)}(\pi_1,\pi_2) = 2(\pi_2 - \pi_1)$, hence supporting our hypothesis (i). $\Omega^{(2,0)}(\pi_1,\pi_2) = \Omega^{(0,2)}(\pi_1,\pi_2) = 2I$ supports the hypothesis (ii). And clearly $\Omega^{(1,1)}(\pi_1,\pi_2) = -2I$ supports the third hypothesis. It can be proved in similar ways that the functional form that we have used for Ω (equation 3) also satisfies the hypotheses (i) to (iii).

To justify the fourth hypothesis we assume that c being the sum of squares of the Euclidean distances between points in the trajectory is a quadratic in π and $c(\pi) \geq 0\ \forall\pi$. Thus the Hessian matrix $c^{(2)}(\pi)$ is a constant matrix, and $\pi^T c^{(2)}(\pi)\,\pi = c(\pi) \geq 0\ \forall\pi$. This hence justifies the condition (iv).

Theoretical Properties for N agents

Similar to before, π_i will indicate a trajectory of robot i passing through its initial and goal configurations. In order to generalize the proof for N agents we generalize the definition for Ψ as follows:

$$\begin{aligned}\Psi(W,\pi_{-r}) &\equiv \Psi(W,\pi_1,\pi_2,\cdots,\pi_{r-1},\pi_{r+1},\cdots,\pi_N)\\ &= \min_{\pi'_r}\left(c(\pi'_r)+\sum_{i,i\neq r} w_{ir}\Omega(\pi_i,\pi'_r)+\sum_{j,j\neq r} w_{rj}\Omega(\pi'_r,\pi_j)\right)\end{aligned}$$

Here, we write π_{-r} to denote the set $\{\pi_1,\pi_2,\cdots,\pi_{r-1},\pi_{r+1},\cdots,\pi_N\}$ for convenience. Also, note that W is a symmetric matrix with non-negative elements and a zero diagonal. The summations are done from 1 to N. For the case of two agents, consistent with this definition, we had, $w = 2w_{12} = 2w_{21}$ and $\Omega = \Omega_{12}$. In the following sections we will use the substitution $\epsilon^k_{i,j} = \epsilon V^k_{i,j}$, where, consistent with our previous definitions, V is a symmetric matrix with non-negative elements and zero diagonal, and

$$V^k_{i,j}\begin{cases}\geq 0, & \forall i = r \text{ or } j = r,\quad i\neq j\\ = 0, & \text{otherwise}\end{cases}$$

Where r is the robot that plans trajectory in the k^{th} iteration. By using this substitution we decouple the *direction* of increment of the penalty weights from the *step-size* of the increments.

Definition 4 (generalized well-behaved cost and distance functions)**.** *The function pair* $\{c,d\}$ *is said to be well-behaved in Graph* G_i *iff there exists small enough* $\epsilon_0 > 0$ *such that for any* $\epsilon_0 \geq \epsilon > 0$ *it holds that*

$$\Psi(W+\epsilon V,\pi^{*W}_{-r}) - \Psi(W,\pi^{*W}_{-r}) \leq \Psi(W+\epsilon V,\pi^{*W+\epsilon V}_{-r}) - \Psi(W,\pi^{*W+\epsilon V}_{-r})$$

$\forall r$*, and for any symmetric matrices* W *and* V *with non-negative elements and zero diagonals. As before, for a given* W *we define*

$$\begin{aligned}\{\pi^{*W}_1,\pi^{*W}_2,\cdots,\pi^{*W}_N\} &= \mathit{argmin}_{\pi'_1,\pi'_2,\cdots,\pi'_N}\Big\{c(\pi'_1)+c(\pi'_2)+\cdots+c(\pi'_N)\\ &\qquad +\sum_{\substack{i,j\\ i\neq j}} w_{ij}\Omega(\pi'_i,\pi'_j)\Big\}\end{aligned}$$

Theorem 3. *Given the condition defined in 4 holds and a sufficiently small* ϵ *is used in the algorithm, every time line 5 is executed the following holds:*

$$\begin{aligned}\{\pi^{iter}_1,\ldots,\pi^{iter}_N\} &= \arg\min_{\pi_1\ldots\pi_N}\textstyle\sum_{i=1\ldots N} c(\pi_i)\\ &\qquad +\textstyle\sum_{j=1\ldots N, j\neq i} w^{iter}_{i,j}\cdot\Omega(\pi_i,\pi_j))\end{aligned}$$

Proof. The theorem clearly holds when line 5 is executed for the first time since $w^0_{i,j} = 0$ for all i,j.

As before, we prove the theorem by induction. Assume it holds for 0th through $iter = 0 \dots k$ executions of line 5. We will now prove that it continues to hold for the $iter = k+1$th execution of line 5. By inductive assumption we have:

$$\{\pi_1^k, \pi_2^k, \cdots, \pi_N^k\} = \arg\min_{\pi_1,\pi_2,\cdots,\pi_N} \left(\sum_i c(\pi_i) + \sum_{\substack{i,j \\ i \neq j}} w_{i,j}^k \Omega(\pi_i, \pi_j) \right)$$

Without loss of generality, let's assume that at k^{th} iteration, the algorithm was computing the path for the r^{th} robot, π_r^{k+1}, and therefore $\pi_j^{k+1} = \pi_j^k, \forall j \neq r$.
We prove by contradiction. Let us assume that

$$\{\pi_1^{k+1}, \pi_2^{k+1}, \cdots, \pi_N^{k+1}\} \neq \arg\min_{\pi_1,\pi_2,\cdots,\pi_N} \left(\sum_i c(\pi_i) + \sum_{\substack{i,j \\ i \neq j}} w_{i,j}^{k+1} \cdot \Omega(\pi_i, \pi_j) \right)$$

This implies that there exist $\pi'_1, \pi'_2, \cdots, \pi'_N$ such that
$\{\pi'_1, \pi'_2, \cdots, \pi'_N\} = \arg\min_{\pi_1,\pi_2,\cdots,\pi_N} \left(\sum_i c(\pi_i) + \sum_{\substack{i,j \\ i \neq j}} w_{i,j}^{k+1} \cdot \Omega(\pi_i, \pi_j) \right)$
and

$$\begin{aligned} \sum_i c(\pi_i^{k+1}) + \sum_{\substack{i,j \\ i \neq j}} w_{i,j}^{k+1} \cdot \Omega(\pi_i^{k+1}, \pi_j^{k+1}) \\ > \sum_i c(\pi'_i) + \sum_{\substack{i,j \\ i \neq j}} w_{i,j}^{k+1} \cdot \Omega(\pi'_i, \pi'_j) \end{aligned} \tag{24}$$

Since the algorithm has just computed π_r^{k+1} it holds that:

$$\pi_r^{k+1} = \arg\min_{\pi_r} \left(c(\pi_r) + \sum_{\substack{i \\ i \neq r}} w_{i,r}^{k+1} \cdot \Omega(\pi_i^{k+1}, \pi_r) + \sum_{\substack{j \\ j \neq r}} w_{r,j}^{k+1} \cdot \Omega(\pi_r, \pi_j^{k+1}) \right)$$

$$\Rightarrow \quad \Psi(W^{k+1}, \pi_{-r}^{k+1}) = c(\pi_r^{k+1}) + \sum_{\substack{i \\ i \neq r}} w_{i,r}^{k+1} \cdot \Omega(\pi_i^{k+1}, \pi_r^{k+1}) + \sum_{\substack{j \\ j \neq r}} w_{r,j}^{k+1} \cdot \Omega(\pi_r^{k+1}, \pi_j^{k+1}) \tag{25}$$

Also, it holds that

$$\pi'_r = \arg\min_{\pi_r} \left(c(\pi_r) + \sum_{\substack{i\\ i\neq r}} w_{i,r}^{k+1} \cdot \Omega(\pi'_i, \pi_r) + \sum_{\substack{j\\ j\neq r}} w_{r,j}^{k+1} \cdot \Omega(\pi_r, \pi'_j) \right)$$

$$\Rightarrow \quad \Psi(W^{k+1}, \pi'_{-r}) = c(\pi'_r) + \sum_{\substack{i\\ i\neq r}} w_{i,r}^{k+1} \cdot \Omega(\pi'_i, \pi'_r) + \sum_{\substack{j\\ j\neq r}} w_{r,j}^{k+1} \cdot \Omega(\pi'_r, \pi'_j) \tag{26}$$

Thus from 24, 25 and 26

$$\sum_{\substack{i\\ i\neq r}} c(\pi_i^{k+1}) + \sum_{\substack{i,j\neq r\\ i\neq j}} w_{i,j}^{k+1} \cdot \Omega(\pi_i^{k+1}, \pi_j^{k+1}) + \Psi(W^{k+1}, \pi_{-r}^{k+1})$$
$$> \sum_{\substack{i\\ i\neq r}} c(\pi'_i) + \sum_{\substack{i,j\neq r\\ i\neq j}} w_{i,j}^{k+1} \cdot \Omega(\pi'_i, \pi'_j) + \Psi(W^{k+1}, \pi'_{-r})$$
$$\Rightarrow \quad \sum_{\substack{i\\ i\neq r}} \left(c(\pi_i^{k+1}) - c(\pi'_i)\right) + \sum_{\substack{i,j\neq r\\ i\neq j}} w_{i,j}^{k+1} \cdot \left(\Omega(\pi_i^{k+1}, \pi_j^{k+1}) - \Omega(\pi'_i, \pi'_j)\right)$$
$$> \Psi(W^{k+1}, \pi'_{-r}) - \Psi(W^{k+1}, \pi_{-r}^{k}) \tag{27}$$

On the other hand, $\{\pi_1^k, \pi_2^k, \cdots, \pi_N^k\}$ is a global minimum for W^k according to our inductive assumption. Using this fact and the hypotheses that $\pi_{-r}^k = \pi_{-r}^{k+1}$ and $w_{i,j}^k = w_{i,j}^{k+1} \forall i \neq r$ and $j \neq r$, similarly it holds that

$$\sum_{\substack{i\\ i\neq r}} c(\pi_i^{k}) + \sum_{\substack{i,j\neq r\\ i\neq j}} w_{i,j}^{k} \cdot \Omega(\pi_i^{k}, \pi_j^{k}) + \Psi(W^{k}, \pi_{-r}^{k})$$
$$< \sum_{\substack{i\\ i\neq r}} c(\pi'_i) + \sum_{\substack{i,j\neq r\\ i\neq j}} w_{i,j}^{k} \cdot \Omega(\pi'_i, \pi'_j) + \Psi(W^{k+1}, \pi'_{-r})$$
$$\Rightarrow \quad \sum_{\substack{i\\ i\neq r}} \left(c(\pi_i^{k+1}) - c(\pi'_i)\right) + \sum_{\substack{i,j\neq r\\ i\neq j}} w_{i,j}^{k+1} \cdot \left(\Omega(\pi_i^{k+1}, \pi_j^{k+1}) - \Omega(\pi'_i, \pi'_j)\right)$$
$$< \Psi(W^{k}, \pi'_{-r}) - \Psi(W^{k}, \pi_{-r}^{k}) \tag{28}$$

Thus, from inequalities 27 and 28, and using $\pi_2^k = \pi_2^{k+1}$,

$$\Psi(W^k, \pi'_{-r}) - \Psi(W^k, \pi_{-r}^k) > \Psi(W^{k+1}, \pi'_{-r}) - \Psi(W^{k+1}, \pi_{-r}^k)$$
$$\Rightarrow \quad \Psi(W^k + \epsilon V, \pi'_{-r}) - \Psi(W^k, \pi'_{-r}) < \Psi(W^k + \epsilon V, \pi_{-r}^k) - \Psi(W^k, \pi_{-r}^k) \tag{29}$$

But the paths $\{\pi_1^k, \pi_2^k, \cdots, \pi_N^k\}$ are the global minimum for W^k by the inductive assumption. From Definition 4 of well-behaved-ness it therefore follows that:

$$\Psi(W^k + \epsilon V, \pi_{-r}^k) - \Psi(W^k, \pi_{-r}^k) \quad < \quad \Psi(W^k + \epsilon V, \pi'_{-r}) - \Psi(W^k, \pi'_{-r}) \tag{30}$$

Thus from the last two inequalities we end up with a contradiction. Hence our assumption of the existence of $\{\pi'_1, \pi'_2, \cdots, \pi'_N\}$ was incorrect.

□

Since theorem 3 basically says that after every iteration the solution paths are cost-minimal w.r.t. the current weight W, the optimality of the algorithm follows from the fact that for a sufficiently large weight W, the weighted penalty function will dominate completely the cost function $c(\cdot)$, and therefore the paths will converge to the cost-minimal path consensus if there exists one. It happens in finite amount of time, since weights are increased by a strictly positive (bounded from below) increment ϵ and assuming there is a bound on the cost of a shortest path (denoted by $maxpathcost$). The following theorem states this formally.

Theorem 4. *Given the condition defined in 4 holds and a sufficiently small ϵ is used in the algorithm, the algorithm terminates, and when it terminates (say at iteration $iter$) the following holds:*

- $\{\pi_1^{iter}, \dots, \pi_N^{iter}\} = \{\pi_1^*, \dots, \pi_N^*\}$ *if there exists any set of paths that minimizes the objective function 1 and satisfies the constraints 2;*
- $\{\pi_1^{iter}, \dots, \pi_N^{iter}\} = \arg\min_{\pi_1 \dots \pi_N} \sum_{i=1\dots N} c(\pi_i) + \sum_{j=1\dots N, j\neq i} w_{i,j}^{iter} \cdot \Omega(\pi_i, \pi_j)$ *otherwise;*

Proof. From result of Theorem 3,

$$\begin{aligned} \sum_i c(\pi_i^k) + \sum_{\substack{i,j \\ i\neq j}} w_{i,j}^k \Omega(\pi_i^k, \pi_j^k) \quad &\leq \quad \sum_i c(\pi_i^{k+1}) + \sum_{\substack{i,j \\ i\neq j}} w_{i,j}^k \Omega(\pi_i^{k+1}, \pi_j^{k+1}) \\ \Rightarrow \quad \sum_{\substack{i,j \\ i\neq j}} w_{i,j}^k \left(\Omega(\pi_i^k, \pi_j^k) - \Omega(\pi_i^{k+1}, \pi_j^{k+1})\right) \quad &\leq \quad \sum_i \left(c(\pi_i^{k+1}) - c(\pi_i^k)\right) \end{aligned} \tag{31}$$

Similarly,

$$\begin{aligned} \sum_i c(\pi_i^k) + \sum_{\substack{i,j \\ i\neq j}} w_{i,j}^{k+1} \Omega(\pi_i^k, \pi_j^k) \quad &\geq \quad \sum_i c(\pi_i^{k+1}) + \sum_{\substack{i,j \\ i\neq j}} w_{i,j}^{k+1} \Omega(\pi_i^{k+1}, \pi_j^{k+1}) \\ \Rightarrow \quad \sum_i \left(c(\pi_i^{k+1}) - c(\pi_i^k)\right) \quad &\leq \quad \sum_{\substack{i,j \\ i\neq j}} w_{i,j}^{k+1} \left(\Omega(\pi_i^k, \pi_j^k) - \Omega(\pi_i^{k+1}, \pi_j^{k+1})\right) \end{aligned} \tag{32}$$

Combining the above two,

$$
\begin{aligned}
& \sum_{\substack{i,j\\ i\neq j}} (w_{i,j}^{k+1} - w_{i,j}^{k}) \left(\Omega(\pi_i^k, \pi_j^k) - \Omega(\pi_i^{k+1}, \pi_j^{k+1}) \right) \geq 0 \\
\Rightarrow \quad & \epsilon \sum_{\substack{i,j\\ i\neq j}} v_{i,j}^k \left(\Omega(\pi_i^k, \pi_j^k) - \Omega(\pi_i^{k+1}, \pi_j^{k+1}) \right) \geq 0 \\
\Rightarrow \quad & \sum_{\substack{i,j\\ i\neq j}} v_{i,j}^k \Omega(\pi_i^k, \pi_j^k) \geq \sum_{\substack{i,j\\ i\neq j}} v_{i,j}^k \Omega(\pi_i^{k+1}, \pi_j^{k+1}) \\
\Rightarrow \quad & V^k \cdot \Omega^k \geq V^k \cdot \Omega^{k+1} \qquad (33)
\end{aligned}
$$

where $\Omega_{i,j}^k := \Omega(\pi_i^k, \pi_j^k)$

We now note that in 33, Ω^k and Ω^{k+1} are non-negative matrices (matrices with non-negative elements). We can choose V^k such that it's elements are non-negative (i.e. a non-negative matrix). This will immediately imply that the sequence $\{|\Omega^1|, |\Omega^2|, \dots\}$ is bounded, and the can also be decreasing under proper choice of V^k.

The algorithm terminates to a feasible solution only when the violation cost becomes zero (step 5 of the algorithm). Then the solution to the problem becomes, by Theorem 3,

$$\{\pi_1^{iter}, \cdots, \pi_N^{iter}\} = \{\pi_1^*, \cdots, \pi_N^*\}$$

Otherwise, termination to an infeasible solution will trivially imply

$$\{\pi_1^{iter}, \cdots, \pi_N^{iter}\} = \arg\min_{\pi_1, \cdots, \pi_N} \sum_i c(\pi_i) + \sum_{\substack{i,j\\ i\neq j}} w_{i,j}^{iter} \cdot \Omega(\pi_i, \pi_j)$$

Now we justify the proposed termination criteria when $\Omega(\pi_1^k, \pi_2^k) \neq 0$ at termination. From result of Theorem 3, and using the fact that $w_{i,j}^k < w_{i,j}^{k+1} = w_{i,j}^k + \epsilon_{i,j}^k$

$$
\begin{aligned}
\sum_i c(\pi_i^k) + \sum_{\substack{i,j\\ i\neq j}} w_{i,j}^k \Omega(\pi_i^k, \pi_j^k) \quad & \leq \quad \sum_i c(\pi_i^{k+1}) + \sum_{\substack{i,j\\ i\neq j}} w_{i,j}^k \Omega(\pi_i^{k+1}, \pi_j^{k+1}) \\
& < \quad \sum_i c(\pi_i^{k+1}) + \sum_{\substack{i,j\\ i\neq j}} w_{i,j}^{k+1} \Omega(\pi_i^{k+1}, \pi_j^{k+1}) \qquad (34)
\end{aligned}
$$

Since we have a discretized space, we can have measurably finite Δ_{ij} such that

$$\Delta_{ij} \quad = \min_{\pi_i, \pi_j} \{\Omega(\pi_i, \pi_j) \text{ s.t. } \Omega(\pi_i, \pi_j) \neq 0\}$$

Then,

$$
\begin{aligned}
& \sum_i c(\pi_i^k) + \sum_{\substack{i,j\\ i\neq j}} w_{i,j}^k \Omega(\pi_i^k, \pi_j^k) \\
& \qquad \leq \sum_i c(\pi_i^{k+1}) + \sum_{\substack{i,j\\ i\neq j}} w_{i,j}^{k+1} \Omega(\pi_i^{k+1}, \pi_j^{k+1}) - \min_{\substack{i,j\\ i\neq j}} \Delta_{ij} \epsilon_{i,j}^k
\end{aligned}
$$

Thus the total cost grows at least linearly with k unless $\Omega(\pi_i^k, \pi_j^k) = 0 \; \forall i, j$. This will guarantee that the infeasibility termination criteria is achieved unless of course a feasible solution is found before that. That is, the value of $\sum_{\substack{i,j \\ i \neq j}} w_{i,j}^k \Omega(\pi_i^k, \pi_j^k)$ does not asymptotically tend to a positive value as k tends to infinity. [Note: Athough the termination criteria is on the violation cost rather than the total cost, we can note that the terms $\sum_i c(\pi_i^k)$ are bounded by $N * maxpathcost$. This will imply that only the violation cost portion is able to increase linearly or faster with k after that bound is reached.])

Next we will prove that if the infeasibility criteria is indeed achieved, then there does not exist a feasible solution.

Termination due to the second condition (infeasibility condition) implies (at the iteration k where the termination takes place),

$$\sum_i c(\pi_i^k) + \sum_{\substack{i,j \\ i \neq j}} w_{i,j}^k \Omega(\pi_i^k, \pi_j^k) > N * maxpathcost$$

and, by Theorem 3,

$$\{\pi_1^k, \cdots, \pi_N^k\} = \arg\min_{\pi_1, \cdots, \pi_N} \left(\sum_i c(\pi_i) + \sum_{\substack{i,j \\ i \neq j}} w_{i,j}^k \Omega(\pi_i, \pi_j) \right)$$

By contradiction, let us assume that there exists a feasible solution $\{\pi_1^{'*}, \cdots, \pi_N^{'*}\}$ so that,

$$\begin{aligned} \{\pi_1^{'*}, \cdots, \pi_N^{'*}\} \quad &= \quad argmin_{\pi_1, \cdots, \pi_N} \sum_i c(\pi_i) \\ &\quad\quad \text{s.t. } \Omega(\pi_i, \pi_j) = 0, \forall i, j \end{aligned}$$

If $\{\pi_i^k, \cdots, \pi_j^k\} \neq \{\pi_i^{'*}, \cdots, \pi_j^{'*}\}$, then by Theorem 3 we should have,

$$\begin{aligned} & \textstyle\sum_i c(\pi_i^k) + \sum_{\substack{i,j \\ i \neq j}} w_{i,j}^k \Omega(\pi_i^k, \pi_j^k) \leq \sum_i c(\pi_i^{'*}) + \sum_{\substack{i,j \\ i \neq j}} w_{i,j}^k \Omega(\pi_i^{'*}, \pi_j^{'*}) \\ \Rightarrow \quad & \textstyle\sum_i c(\pi_i^k) + \sum_{\substack{i,j \\ i \neq j}} w_{i,j}^k \Omega(\pi_i^k, \pi_j^k) \leq \sum_i c(\pi_i^{'*}) \\ & \qquad\qquad\qquad\qquad\qquad\qquad [\because \Omega(\pi_i^{'*}, \pi_j^{'*}) = 0] \end{aligned} \tag{35}$$

Thus, by using the termination criteria we have,

$$\sum_i c(\pi_i^{'*}) \quad > \quad N * maxpathcost$$

Which is not possible in a discrete graph of finite size. Thus there can't exist $\{\pi_1^*, \cdots, \pi_N^*\}$ as hypothesized.

□

```
1 procedure DPC()
2 set $iter = 0$;
3 set $BLIST^{iter} = \emptyset$;
4 while $(\sum_{i=1\dots N} \sum_{j=i+1\dots N} \Omega(\pi_i^{iter}, \pi_j^{iter}) \neq 0)$
5     run $BLIST^{new} = BasicDPC_{modified}(BLIST^{iter})$;
6     $BLIST^{iter+1} = BLIST^{iter} \bigcup BLIST^{new}$;
7     $iter = iter + 1$;
```

Figure 3: DPC with superiterations

DPC Algorithm with Superiterations

The DPC algorithm without superiterations typically converges to an optimal solution in an empty environment. The space filled with obstacles, however, presents an environment in which small perturbations to the penalty weights can cause large jumps in the path from one homotopy classes of paths to another. This not only tend to result in suboptimal solutions (if a feasible solution is found), but may also need us to increase the penalty weights by large values before the transition from an infeasible homotopy class can take place to a feasible one. This presents a problem to the algorithm presented in the previous section since it relies on the ability to slowly vary the trajectories to satisfy the distance constraints by gradually increasing pathcosts. As shown in figure 4 example, obstacles can be positioned in such a configuration that the trajectories get trapped in a infeasible homotopy class and cannot get modified by the DPC algorithm without superiterations. To resolve this and to guarantee completeness, we gradually build a list, called $BLIST$ (short for *blacklist*) of pairwise configurations in the joint state-space that are invalid.

Each entry in the blacklist is of the following format:

$$BLIST[l] = \{\{i, j\}, t, \{\mathbf{s}_i, \mathbf{s}_j\}\}$$

where i and j are indices of the robots that violate a constraint in between them at time t (namely, the constraint specified by $\phi_{i,j}(t)$), and $\mathbf{s}_i \in \mathbf{V}(G_i)$ and $\mathbf{s}_j \in \mathbf{V}(G_j)$ are corresponding states of these robots that violate this constraint.

The pseudocode of the algorithm is shown in figure 3. It first initializes the list $BLIST$ to an empty set and then repeatedly calls a modified version of the basic DPC algorithm ($BasicDPC_{modified}$) passing in the current $BLIST$ 5.

The modified version of the $BasicDPC$ operates exactly as before (figure 1) except with three modifications:

i. Within each graph search, avoids the configurations within a radius of α of the configurations in the $BLIST^{iter}$ that are known to be invalid. When search within an iteration of $BasicDPC_{modified}$ computes a path π_r^{iter} for the robot R_r, if $\{\{r, j\}, t^0, \{\mathbf{p}, \mathbf{q}\}\}$ is an entry in the blacklist, and robot r is planning its trajectory at $(iter+1)$ iteration, it will consider the state (i.e. a cell in the space-time planning environment) $\{\mathbf{s}, t^0\}$ to be an obstacle (i.e. an invalid cell for the graph H_r) if $\sqrt{|\mathbf{s}-\mathbf{p}|^2 + \left|\pi_j^{iter}(t^0) - \mathbf{q}\right|^2} < \alpha$. Thus, basically it avoids a 4 dimensional ball of radius α around the blacklisted point in the joint statespace of robots r and $j(\neq r)$. If we choose α small enough such that it is small compared to the thickness or radius of all obstacles in the environment, this method will gradually eliminate infeasble homotopic classes and will converge to a feasible solution.

ii. There is an additional termination criteria that will detect consistent lack of change in the trajectories violating the constraints in spite of increasing the weights significantly. This will indicate that the present homotopic class of trajectories is infeasible and hence need to terminate that particular superiteration.

iii. After every graph search finishes, the $BasicDPC_{modified}$ function processes the path that was just computed and finds all the constraints it violates. All of these violations are being added to $BLIST_{new}$, which is returned from the $BasicDPC_{modified}$ function. Suppose in one of the iterations inside $BasicDPC_{modified}$ robot R_r was planning. Then, in that iteration if there has been a violation of a constraint such that $\omega(\pi_r(t^0), \pi_j(t^0), \phi_{r,j}(t^0)) > 0$ for some j, t^0, the algorithm adds $\{\{r,j\}, t^0, \{\pi_r(t^0), \pi_j(t^0)\}\}$ to $BLIST_{new}$.

Example 2. *Figure 4 demonstrates the operation of the DPC Algorithm with superiterations. The environment is also of size 1 unit by 1 unit discretized into 100 by 100 cells. All the robots start at time* $t = 0$ *units and need to reach their goals at no later than* $t = 8$ *time units. Time is again discretized at the resolution of* 0.1 *units. The constraint in between the robot 1 (the left-most robot) and the robot 2 (the right-most robot) is that the distance in between them needs to be less than 0.2 units at time* $t = 4.0$*, and can be anything else at other times. Also, the constraint in between the robot 1 (the left-most robot) and the robot 3 (robot in the middle) is that the distance in between them needs to be less than 0.02 units at time* $t = 4.0$*, and can be anything else at other times. The first superiterations, which corresponds to the operation of the* $BasicDPC_{modified}$*, fails to find a feasible solution. The constraints that are violated are between* R_1 *and* R_2 *(left-most and right-most robots). The solution found by robots 1 and 3 are the ones lying to the left of the circular obstacle, whereas the feasible solution requires their trajectories to be on the right side of the circular obstacle. Because these two solutions belong to two different homotopy classes of trajectories, it is impossible to deform one into the other gradually. DPC with superiterations gradually builds the blacklist ruling out the invalid configurations in the joint state-space of the robots violating the constraints (*R_1 *and* R_2*). Thus, figure 4(b) shows that the blacklist contains one single entry* b_1 *- a 4-dimensional ball indicating a joint state for robots 3 and 2 at the time of the constraint. This list is being grown after every superiterations. At the end of the 8th superiteration, the trajectories of the robots 1 and 3 switch to the right of the large circular object and the globally feasible solution is found.*

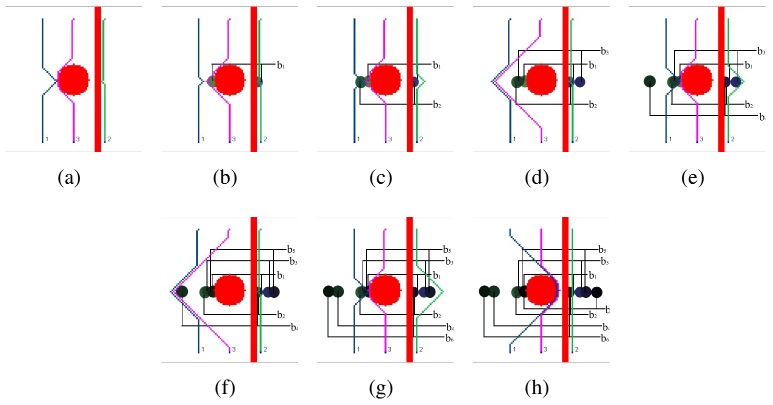


Figure 4: Planning in environment of example 2 using DPC with superiterations. Note that the pairs of black circles, $\mathbf{b}_i$ in the figure are 2-D projections of the 4-D blacklisted balls in the joint state-space $H_1 \times H_2$

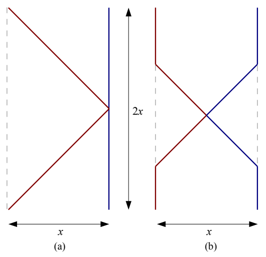


Figure 5: Two possible solutions for two robots in an 8-connected grid

Results

Simulation in a Discretized Environment

Discretization

This three-dimensional state-space of each robot, H_i, was discretized into uniform cells, each cell representing nodes of the search graph. The connectivity of the graph was such that a cell in the x, y, t space connected to its 8 neighboring cells and itself in x, y but with the time index incremented by one. While an 8-connected grid is quick and efficient to perform search in, it confines the motion of the robot to 8 directions ($45°$ orientations). A consequence of this is that some seemingly sub-optimal solutions are actually optimal.

For example, consider two robots separated by a distance of x units. They need to reach goals that are $2x$ units of distance away from them. Also, suppose that a constraint requires them to meet in the middle of their paths. In an 8-connected grid the two possible solutions that the robots can take are shown in figure 5.

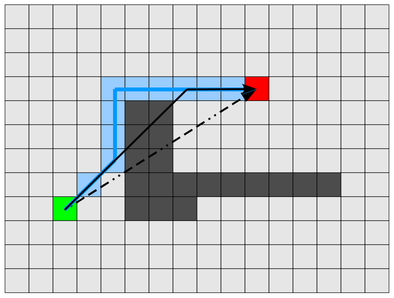


Figure 6: The three different types of heuristics for 8-connected grids: The green is the start state and red is the goal state. Dashed arrow represents h_E, solid black arrow represents h_8, light blue path represents the one from the Dijkstra search. Obstacles are in gray.

It is easy to see that both sets of paths, (a) and (b), are valid solutions to the problem. While seemingly very different however, in both of these solutions, the summations of the costs of the paths are the same and are equal to $2(1+\sqrt{2})x$. This implies that, since our objective function is the summation of the costs of the paths, both of these outcomes are equally feasible and optimal in an 8-connected graph.

Heuristic

For each graph search (corresponding to the discretized 3-dimensional space), we used A* algorithm [24]. The choice of the heuristic function, h, is extremely crucial in any A* search. A heuristic function is a positive scalar function of the states in the search graph. For an A* to return an optimal solution, an admissible heuristic function needs to be such that it never overestimates the actual minimum cost for reaching the goal. However for making the search more efficient so that the least number of states are expanded, the heuristic must as close as possible to the actual minimum cost to the goal. One obvious and commonly used heuristic for planning in G_i is the Euclidean distance to the goal, i.e. $h_E(\mathbf{s}) = \|\mathbf{s} - \mathbf{s}_{goal}\|$. But for an 8-connected grid, which G_i is, one can use a more efficient heuristic given by $h_8(\mathbf{s}) = \sqrt{2}\min(\Delta x, \Delta y) + |\Delta x - \Delta y|$, where $\Delta x = |\mathbf{s}_x - \mathbf{s}_{goal,x}|$ and $\Delta y = |\mathbf{s}_y - \mathbf{s}_{goal,y}|$.

However in an environment with plenty of obstacles even h_8 turns out to be highly inefficient. In such a case, before executing the actual planning, we perform a Dijkstra's search in G_i starting from the goal coordinate $\mathbf{s}_{goal}$ till we expand all the accessible states in G_i. Let the cost associated with each expanded state (starting from $\mathbf{s}_{goal}$) as obtained from Dijkstra search be $D_{\mathbf{s}_{goal}}(\mathbf{s})$. Although the process of Dijkstra search is itself more expensive than a simple A* search in G_i, the advantage of this pre-computation becomes clear when we attempt to plan in higher dimensional graphs H_i. Once D is precomputed for all the states in G_i, while planning in H_i we can simply use the more efficient heuristic function, $h_H(\{\mathbf{s}, t\}) = D_{\mathbf{s}_{goal}}(\mathbf{s})$. Figure 6 illustrates the different heuristic functions.

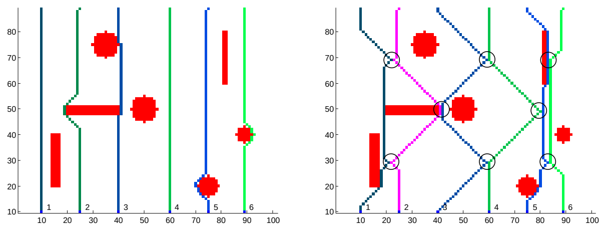

(a) Unconstrained trajectories (b) Converged feasible solution

Figure 7: Planning in a simple environment

In the following subsections, we will present the performances of our algorithm in different environments. [2]

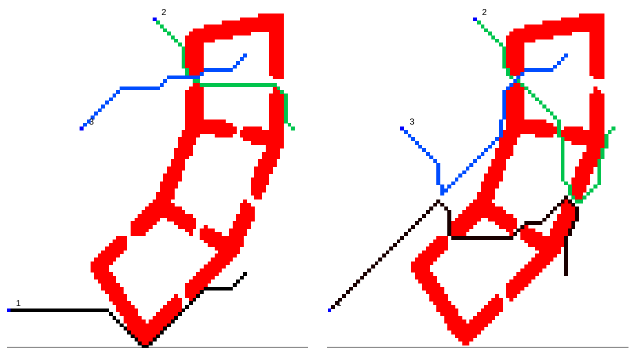
A simple environment

This environment shown in Figure 7 is similar to Environment of Example 1 consisting of 6 robots in an environment with 7 obstacles. The environment discretization and the constraints are exactly same as that described in Example 1. Some of the obstacles in the environment were intentionally placed so as to disrupt the relatively easy rendezvous of the robots as they were done in the environment without obstacles. Figure 7(a) shows the unconstrained planning done by the individual robots in the environment. The iterations converged to a feasible solution within the very first superiteration, despite the presence of obstacles. We set the penalty weight increment to $\epsilon = 0.1$. The number of iterations required for the convergence was 25, and the time taken for the overall planning was less than 20 seconds. Figure 7 shows the final converged solution. The rendezvous points are marked by the black circles.

Computation time

We ran multiple simulations in a downsampled version of the environment (with obstacles) of Figure 7 with 500,000 states (50×50 cells along the spacial directions and 200 cells along the time) and 6 robots. In each run we randomized the constraints and noted the time required to do the complete planning. The minimum planning time was 117 seconds and the maximum planning time was 142 seconds, with an average time of 129.6 seconds. We note that the joint state-space of the robots has $200 \times (50 \times 50)^6 = 4.883 \times 10^{22}$ states. Finding a plan that satisfies all the constraints in such a huge state-space is itself an extremely challenging task. Our algorithm not only does the planning substantially fast, but also has the potential of guaranteeing convergence, completeness and optimality (we have presented the proofs for the case of 2

[2]All the corresponding output files and the matlab scripts for viewing the output files can be found at http://www.seas.upenn.edu/ subhrabh/nonWebsite/IterPlanning/index.html.



(a) Unconstrained plans (b) Converged feasible solution

Figure 8: Planning in environment with three interconnected rooms

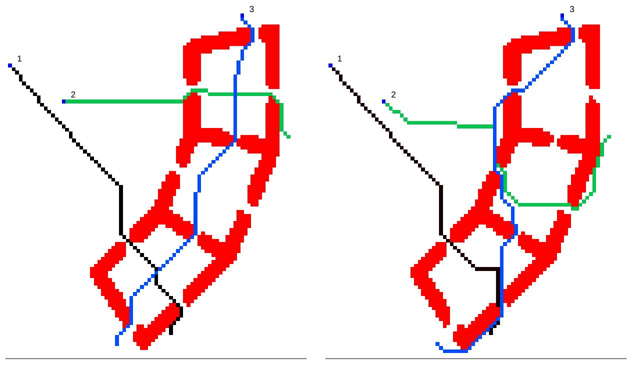
robots in an empty environment. We are working towards generalizing the proofs for more than 2 robots in environments with obstacles.).

Three interconnected rooms

This environment shown in Figure 8 consists of three interconnected rooms as shown. The size of the environment is 11.125 by 11.75 units and discretization size is 0.125 units. The size of the environment in time is 5 units with discretization of 0.025 units. That results in the graph $H_i, \quad i = 1, 2, 3$ to have almost 2 million states. The robots start at $t = 0$ from the left side of the environment and need to reach their goals on the right end latest by $t = 4.0$. Figure 8(a) shows the unconstrained optimal plan. A constraint is defined between R_1 (black) and R_3 (blue) such that they need to be within a distance of 0.3 units at $t = 1.0$. And the constraint between R_1 (black) and R_2 (green) is such that they need to be within a distance of 0.3 units at $t = 3.0$. There is no other distance constraint. With a penalty weight increment of $\epsilon = 0.1$ the algorithm converges to the solution (Figure 8(b)) in about 60 iterations within the very first superiteration.

Extended randevouz

This environment shown in Figure 9 is very similar to the previous example. The size and discretization are exactly the same as before. In this example R_1 (black) and R_2 (green) need to traverse the environment from left to right, while R_3 (blue) needs to traverse it from top to bottom. Figure 9(a) shows the unconstrained optimal plan. Constraints between R_2 and R_3 is that they need to be within 0.3 units distance from $t = 0.8$ to $t = 1.7$ (36 time steps), and the constraints between R_1 and R_3 is that they need to be within 0.3 units distance from $t = 2.1$ to $t = 3.0$ (36 time steps). There is no other distance constraint. With a penalty weight increment of $\epsilon = 0.1$ the



(a) Unconstrained plans (b) Converged feasible solution

Figure 9: Extended randevouz in environment with three interconnected rooms

algorithm converges to the solution (Figure 8(b)) in the second superiteration (about 100 iterations/searches in total).

Extended randevouz in a real environment

This environment shown in Figure 10 is the map of a part of the fourth floor of the Levine hall in University of Pennsylvania that includes the L457 room. The map is 35 meters by 35 meters, discretized into 100x100 cells (each cell 35cmx35cm, almost the dimension of the Scarab robots, described later). The unconstrained objectives, resulting in the solution in Figure 10, are as follows:

i. Robot 1 is to start at t=0.0 inside the big room at the left side of it. It needs to reach the smaller room at the top and the one on the left, by t=170.0. Thus robot 1 traverses the map from bottom to the top.

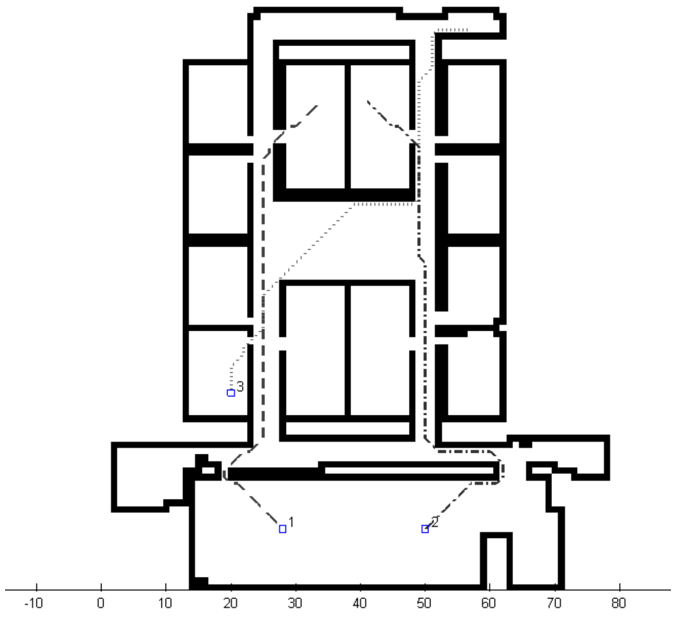
ii. Robot 2 needs to start at t=0.0 inside the big room at the right side of it. It needs to reach the smaller room at the top and the one on the right, by t=170.0. Thus robot 2 traverses the map from bottom to the top.

iiii. Robot 3 needs to start at t=45.0 inside the small lower cubicle on the left side of the map. It needs to reach the smaller storage space at the top right of the map, by t=120.0. Thus robot 3 moves diagonally and traverses the map from left to right.

Figures 11 shows the converged feasible solution that satisfy the following constraints:

A. Robot 2 first needs to meet robot 3 at t=60.0 and needs to stay within 50cm of it for a period of 20 time steps (till t=80.0).

B. Robot 1 then needs to meet robot 3 at t=90.0 and needs to stay within 50cm of it for a period of 20 time steps (till t=110.0).



(a) Unconstrained plans

Figure 10: Map of fourth floor of Levine hall including room L457

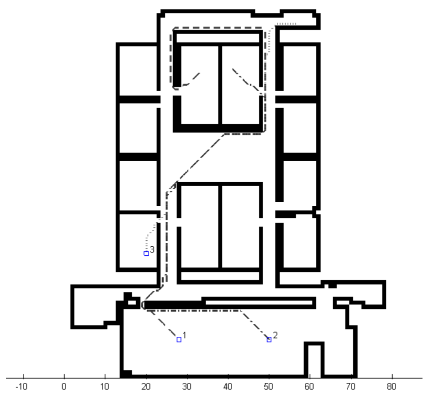
Looking at the converged solution we observe that in order to satisfy constraint A, robot 2 loops it's trajectory around the central and lower cubicles to meet up with robot 3 on the left side of the map from t=60.0 to t=80.0. Then it follows the shortest past to its goal. Again, in order to satisfy constraint B, robot 1 continues moving along with robot 3 for a while and hence ends up moving a little excess to the north. Once the meeting from t=90.0 to t=110.0 is done, the robot 1 takes the shortest path to its goal from there, because of which it loops its trajectory around the central and upper cubicles.

Using a visualization tool we generated an animation movie demonstrating execution of the plan. Collision between the robots and with the walls were avoided using a local greedy collision check alogorithms. The movie can be found at http://www.seas.upenn.edu/ subhrabh/nonWebsite/IterPlanning/index.html.

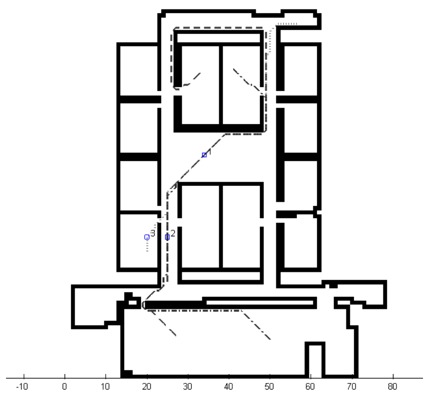
Simulation in Gazebo

Gazebo is a open-source multi-robot simulator for outdoor environments. It is capable of simulating a population of robots, sensors and objects in real time and does so in a three-dimensional world. It generates both realistic sensor feedback and physically plausible interactions between objects. It includes an accurate simulation of rigid-body physics and makes use of the Open Dynamics Engine to do so. [1]

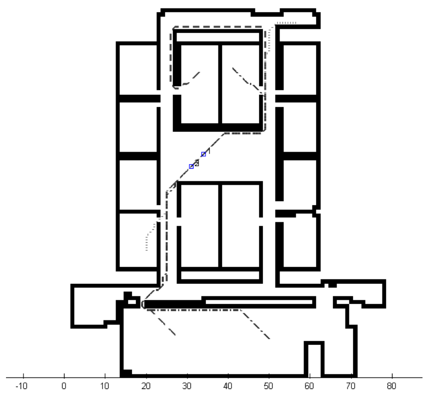
We tested our algorithm on the environment in Example 2 in Gazebo simulated en-



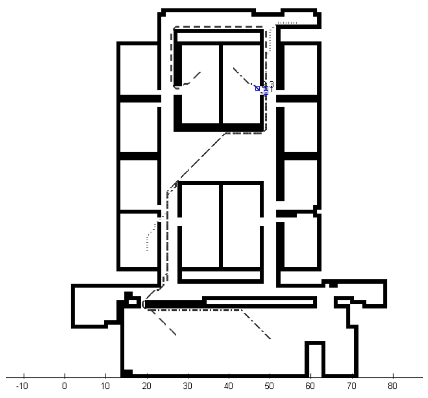
(a) t=1



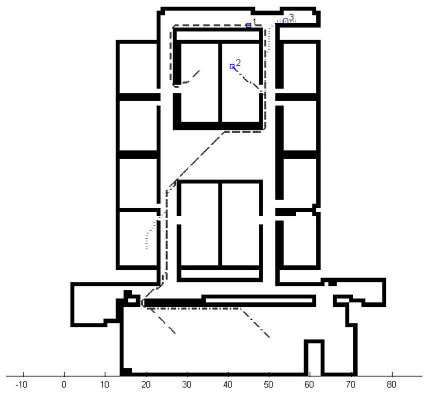
(b) t=50



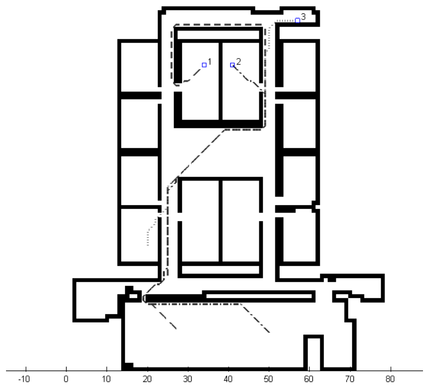
(c) t=67



(d) t=95

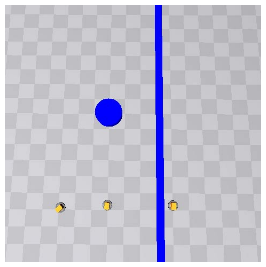


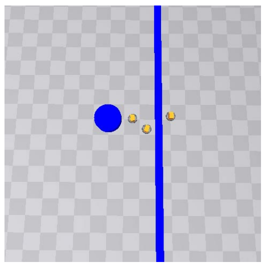
(e) t=115



(f) t=171

Figure 11: Demonstration of extended randevouz showing screenshots at different time instants


(a) Screenshot 1


(b) Screenshot 2

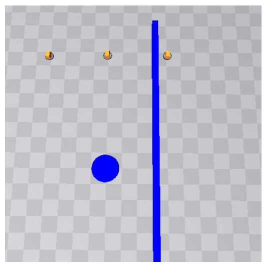

(c) Screenshot 4

Figure 12: Screenshots from Gazebo simulation

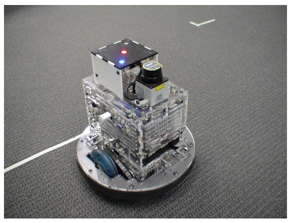
vironment and on real robots. In order to account for the non-zero radii of the robots, we performed a greedy collision avoidance during run-time. For controlling the non-holonomic robots, a feedback linearization technique was adopted [13]. Some screenshots from the Gazebo simulation are shown in Figure 12. They show the execution of the plan generated for 3 robots by the DPC with superiterations algorithm as shown in the figure 4(h). The full movie of the simulation can be found at [10].

Experiment with real Mobile Robots

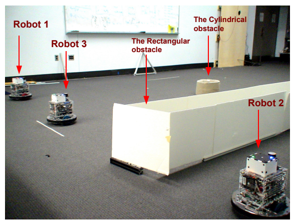
Experiment was performed with mobile robots as well. The mobile robot platforms named *Scarab* were used along with an overhead LED tracking system to track the positions of the robots. The platforms run using robot device interfaces provided by Player [2]. Figure 13(a) shows the Scarab mobile robot platform. Figure 13(b) shows a screenshot of the experiment that was run in an indoor setup of Figure 4. The full movie of the experiment can be found at [10].

Conclusions

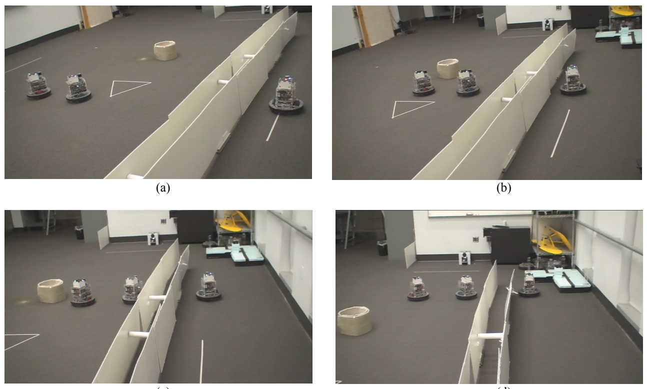
While planning with discrete graph searches has proven to be successful for single-agent tasks, planning for teams of coupled robots is hard due to the high dimensionality of the joint state-spaces. The algorithm we presented in this paper avoids planning in these high-dimensional spaces by breaking the planning problem into a series of low-dimensional planning problems, each being a problem of planning a trajectory in the workspace of a single robot augmented with an additional dimension, namely time. Thus, each search is fast and the overall planning requires little memory. In addition, the algorithm lends itself well to distributed implementation, in which each iteration of the planning can be done on each of the robots separately, and the only required inter-communications are the generated paths and the contents of the blacklist. We have shown analytically that the algorithm guarantees convergence, completeness and optimality in an empty environment under certain conditions.



(a) Scarab - the mobile robot platform



(b) Setting up the environment in an indoor environment



(c) Screenshots from the experiment

Figure 13: Real robot experiments on the environment of Figure 4

References

[1] Gazebo.

[2] The player project.

[3] P. Abichandani, H. Y. Benson, and M. Kam. Multi-vehicle path coordination under communication constraints. In *American Control Conference*, pages 650–656, June 2008.

[4] Ali Ahmadzadeh, Gilad Buchman, Peng Cheng, Ali Jadbabaie, Jim Keller, Vijay Kumar, and George Pappas. Cooperative control of uavs for search and coverage. *Proceedings of the AUVSI Conference on Unmanned Systems*, 2006.

[5] Ali Ahmadzadeh, James Keller, George J. Pappas, Ali Jadbabaie, and Vijay Kumar. Critical cooperative surveillance and coverage with unmanned aerial vehicles. In Vijay Kumar Oussamma Khatib and Daniela Rus, editors, *International Symposium on Experimental Robotics*, STAR, Rio de Janeiro, July 2006. Springer-Verlag.

[6] Daniel S. Bernstein, Eric A. Hansen, and Shlomo Zilberstein. Bounded policy iteration for decentralized pomdps. *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1287–1292, 2005.

[7] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2007.

[8] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation:Numerical Methods*. Prentice Hall, 1989.

[9] D.P. Bertsekas and J.N. Tsitsiklis. Some aspects of parallel and distributed iterative algorithms - a survey. *Automatica*, 27(1):3–21, 1991.

[10] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay Kumar. Files related to this paper, 2009.

[11] Thomas Justin Chan and Candace Arai Yano. A multiplier adjustment approach for the set partitioning problem. *Operations Research*, 40(S1):40–47, 1992.

[12] David C Conner, Alfred Rizzi, and Howie Choset. Composition of local potential functions for global robot control and navigation. In *Proc. Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 3546–3551, 2003.

[13] B. d'Andrea Novel, G. Campion, and G. Bastin. Control of nonholonomic wheeled mobile robots by state feedback linearization. *The International Journal of Robotics Research*, 14(6), Sept. 1995.

[14] J. Desai, J. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 17(6):905–908, December 2001.

[15] Marie E. desJardins, Edmund H. Durfee, Jr. Charles L. Ortiz, and Michael J. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, 2000.

[16] M Bernardine Dias, Robert Michael Zlot, Nidhi Kalra, and Anthony (Tony) Stentz. Market-based multirobot coordination: a survey and analysis. *Proc. of the IEEE*, 94(7):1257 – 1270, 2006.

[17] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[18] Edmund H. Durfee. Distributed problem solving and planning. *Lecture Notes in Computer Science*, 2086(2):118–149, 2001.

[19] E. Ephrati and J. Rosenschein. The clarke tax as a consensus mechanism among automated agents. In *Proceedings of AAAI-91*, pages 173–178, 1991.

[20] Eithan Ephrati, Martha E. Pollack, and Jeffrey S. Rosenschein. A tractable heuristic that maximizes global utility through local plan combination. In *Proceedings of the 1st International Conference on MultiAgent Systems (ICMAS-95)*, 1991.

[21] M. L. Fisher, R. Jaikumar, and L. N. Van Wassenhove. A multiplier adjustment method for the generalized assignment problem. *Management Science*, 32(9):1095–1103, 1986.

[22] Brian Gerkey and Maja Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *Int'l. J. of Robotics Research*, 23(9):939–954, 2004.

[23] Eric A. Hansen and Rong Zhou. Anytime heuristic search. *Journal of Artificial Intelligence Research (JAIR)*, 28:267–297, 2007.

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

[25] N. Jennings. Controlling cooperative problem in industrial multi agent systems using joint intentions. *Artificial Intelligence*, 75(2):195–240, 1995.

[26] Nejat Karabakal and James C. Bean. A multiplier adjustment method for multiple shortest path problem. Technical report, The University of Michigan, June 1995.

[27] R. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, 1990.

[28] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[29] M. Likhachev and D. Ferguson. Planning long dynamically-feasible maneuvers for autonomous vehicles. In *Proceedings of Robotics: Science and Systems (RSS)*, Cambridge, USA, June 2008.

[30] M. Likhachev, G. Gordon, and S. Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems (NIPS) 16*. Cambridge, MA: MIT Press, 2003.

[31] Chien-Chou Lin, Chi-Chun Pan, and Jen-Hui Chuang. A novel potential-based path planning of 3-d articulated robots with moving bases. *Robotica*, 22(4):359–367, 2004.

[32] Savvas G. Loizou and Kostas J. Kyriakopoulos. Closed loop navigation for multiple holonomic vehicles. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2861–2866, 2002.

[33] Nathan Michael, Michael M. Zavlanos, Vijay Kumar, and George J. Pappas. Maintaining connectivity in mobile robot networks. In *International Symposium on Experimental Robotics (ISER)*, July 2008.

[34] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. *18th International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[35] I. Necoara and J.A.K. Suykens. Interior-point lagrangian decomposition method for separable convex optimization. *Journal of Optimization Theory and Applications*, 143:567–588, 2009.

[36] Giuseppe Notarstefano, Ketan Savla, Francesco Bullo, and Ali Jadbabaie. Maintaining limited-range connectivity among second-order agents. In *Proceedings of the 2006 American Control Conference*, June 2006.

[37] Guilherme Augusto Silva Pereira, Aveek K. Das, Vijay Kumar, and Mario F. M. Campos. Decentralized motion planning for multiple robots subject to sensing and communication constraints. In *Proceedings of the Second MultiRobot Systems Workshop*, 2003.

[38] Martha E. Pollack. Planning in dynamic environments: The "dipart"" system. *Advanced Planning Technology: Technology Achievements of the ARPA/Rome Laboratory Planning Initiative*, 1996.

[39] S. Quinlan and O. Khatib. Elastic bands: connecting path planning and control. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 802–807, May 1993.

[40] E. Rimon and D.E. Koditschek. The construction of analytic diffeomorphisms for exact robot navigation on star worlds. *Trans. of the American Mathematical Society*, 327(1), Sept. 1991.

[41] Sikandar Samar, Stephen Boyd, and Dimitry Gorinevsky. Distributed estimation via dual decomposition. 2007.

[42] Matthijs T.J. Spaan, Geoffrey J. Gordon, and Nikos Vlassis. Decentralized planning under uncertainty for teams of communicating agents. *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 249–256, 2006.

[43] A. Stentz. The focussed D* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1652–1659, 1995.

[44] A. Stentz and M. Hebert. A complete navigation system for goal acquisition in unknown environments. *Autonomous Robots*, 2(2):127–145, 1995.

[45] P. Svestka and M. H. Overmars. Probabilistic path planning. Technical Report UU-CS-1995-22, Department of Information and Computing Sciences, Utrecht University, 1995.

[46] William Yeoh, Ariel Felner, and Sven Koenig. Bnb-adopt: An asynchronous branch-and-bound dcop algorithm. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 591–598, 2008.

[47] Milos Zefran. *Continuous methods for motion planning*. PhD thesis, University of Pennsylvania, Philadelphia, PA, USA, 1996. Supervisor-Vijay Kumar.

[48] H. Zhang, V. Kumar, and J. Ostrowski. Motion planning under uncertainty. In *IEEE International Conference on Robotics and Automation*, Leuven, Belgium, May 16-21 1998. IEEE.