Biologically Inspired Algorithms for Optimal Control

by Cheng Shao and D. Hristu-Varsakelis

TR 2004-29

Biologically Inspired Algorithms for Optimal Control[1]

Technical Report

Cheng Shao and D. Hristu-Varsakelis
Department of Mechanical Engineering & Institute for System Research
University of Maryland
College Park, MD 20742
{cshao,hristu}@glue.umd.edu

[1]This work was supported by the National Science Foundation under Grant No. EIA0088081 and by ARO ODDR&E MURI01 Grant No. DAAD19-01-1-0465, (Center for Communicating Networked Control Systems, through Boston University).

Contents

Chapter 1

Introduction

Cooperative control systems are increasingly emerging as significant alternatives to their centralized counterparts recently. The rising interest in deploying cooperative systems is fueled by the development of decentralized systems with low cost and performance advantages. For example, mobile exploration and information gathering tasks can often be accomplished

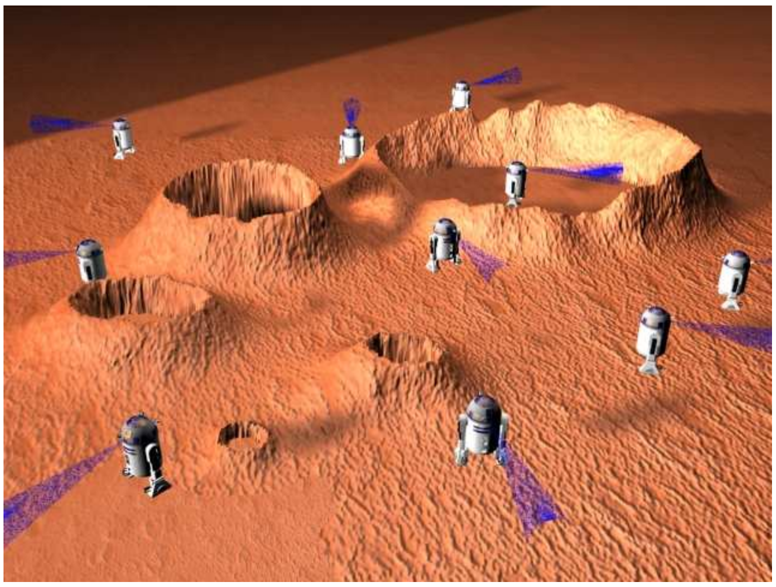


Figure 1.1: A swarm of robots are expected to explore unknown planets.

cheaply and more reliably by swarms of small autonomous robots as opposed to a single more sophisticated one. Cooperative control is also applied in many tasks that can not be performed by a single system, e.g. satellite arrays that enable global communication, geographically remote systems that communicate via network and others.

The goal of our research is to investigate optimal control in cooperative systems, using algorithms inspired from biology. We begin with a review of collective behavior in biological systems.

1.1 Cooperative Biological Systems

Animal aggregation is a common phenomenon in nature, seen in organisms that range in complexity from primal zooplanktons to advanced mammals. Many species exhibit collective movement patterns which are highly organized, compared to the seemingly random individual behaviors. For example, a school of fish can move together in a tight formation and respond almost as fast as a single organism to evade encountering dangers. Worker honey bees can distribute themselves to different nectar sources in accordance with the profitability of each source. Ants can recruit their nest-mates to form a trail along the most efficient path between the nest and food when foraging [1, 2].

The above examples show that aggregate behaviors in these animals may have special group-level properties that go beyond the ability of an individual. Certainly, if all group members' behaviors are coordinated by a centralized "leader", the leader must have the capabilities to communicate with others and alter their behaviors. Observing the qualitatively identical behaviors of all members in an insect aggregate as well as their physical limitations, we can conclude that there are no such leaders in these groups (and this is supported by other research [1, 2, 4]). Therefore, some of the awe-inspiring group behaviors in nature come about as the results of individuals' self-organized actions. For instance, at the individual level honey bees receive limited information from other workmates and go to forage the selected flowers. This type of behavior seems to lead to random distribution over different sources because the message each bee obtained does not convey to it accurate information about the profitability of each nectar source. At the group level, however, it is amazing to see that foragers are rationally dispatched over different flowers in accordance with the distribution of nectar over various sources. Coordinating a colony of bees to achieve such a complicated collective behavior seems very difficult for any individual bee. A reasonable explanation is that bees only follow some simple rules all through the foraging activities while the collective behavior turns out to be highly organized [1]. In conclusion, the individual behavior is an "unsophisticated" one due to the individuals' physical limitations, in contrast to the complex performance of the whole group. This fact implies that there seems to exist an intrinsic mechanism among insect aggregates that overcomes individuals' drawbacks and yields results that might be impossible for individuals to attain. It is the cooperation between group members, i.e. the rule that each individual complies with, that yields group patterns[1] qualitatively different from and more elegant than those of individual behaviors.

[1]We use "group pattern" to refer to the collective movement pattern of a group.

We have seen that biological systems, especially social insects, demonstrate many promising cooperative solutions to complicated tasks. Many of these tasks are similar (at least functionally) to what one might want to do with cooperative engineered collectives. In addition, individual members in a biological collective are similar to the units of a cooperative control system in the sense that they are equipped with limited capabilities of sensing, communicating and computing. What we are essentially interested in is trading off individual capability of cooperating in order to achieve a complex task with less sophisticated equipment: low power, short sensing range and low communication burden, looking to natural examples - like that ants are able to find the most efficient path while individual ants are of short sight and low intellect - for successful prototypes. Natural systems have developed such capabilities to solve various problems through evolution and natural selection, and may offer us some clues on how to proceed [1, 26].

1.2 Research Objectives

The objective of this work is to investigate the cooperative solution of a class of optimal control problems using groups of agents[2] with limited sensing and computing capabilities. Our approach will be to postulate rules for individual behavior, inspired from observations of biological systems, and then investigate the "group pattern" that emerges. Rules for individual agents will be obtained by:

1. Constructing a proper model for the observed collective movement patterns of certain biological systems, including ant colonies. An effective model will allow us to capture some aspects of the "experience" accumulated through natural selection.

2. Extracting simple "rules" that capture individual behavior within the group. These rules should be kept simple, with respect to the computation and communication resources required to implement them, to be applied to cooperative control systems, such as cheap autonomous robots.

3. Exploring how these rules can be applied to artificial collectives in order to solve optimization control problems that are hard or impossible for an individual to solve. This will involve combing existing methods on optimal control with the specified rules.

[2]Throughout the document we will use "agent" to refer to a member of a group of control systems.

1.3 Outline

The rest of this paper is organized as follows: in Chapter 2 we will review various recent research directions of cooperative systems. A class of algorithms for cooperative optimal control inspired from the observed movements of ant colonies will be introduced in Chapter 3, along with a discussion of the algorithms' potential advantages. Chapter 4 presents some current progress concerning the proposed algorithms, including convergence analysis, special cases and numerical experiments. Finally, the ongoing work and an outline of possible approaches for its completion are given in Chapter 5.

Chapter 2

Literature Review

The potential of a cooperating group to "do better than the sum of its parts" has already seeded a variety of recent research directions in engineering, from modeling of animal groups [1, 24, 25, 26], to distributed collective covering and searching [28, 29], estimating by groups [30, 31, 32], cooperative robotic teams [33, 34, 42] and biologically-motivated optimization [36, 27]. These works typically treat narrowly defined problems [36, 32, 27], discuss only the feasibility of special tasks [33, 34, 42], or show the effectiveness instead of optimality of various proposed algorithms [28, 29]. Here, we review some of these and other relevant works.

2.1 Animal Group Pattern Modeling

The work of [24] proposed a simple model concerning the movement of n autonomous agents with the same speed but with varying headings. If each agent of a group uses the "nearest neighbor rule" to update its heading, that is

$$\langle \theta_i(t) \rangle_r = \frac{1}{1 + n_i(t)} \left(\theta_i(t) + \sum_{j \in \mathcal{N}_i(t)} \theta_j(t) \right)$$

where $\theta_i(t)$ is the heading of the i^{th} agent and $n_i(t)$ is the number of neighbors of the i^{th} agent at time t, then all agents' headings will converge to a common constant vector as time goes on. The theoretical explanation for the convergence described in the above model is provided in [25], along with several similar models inspired by [24], such as "leader following" showing that if there exists an agent acting as the "leader" in the group, all agents will evolve to point to the same heading as the leader . This "nearest neighbor rule" can cause all the members of a group to move towards the same direction despite the fact that there is no centralized coordination and that an agent's set of nearest neighbors might change as the system evolves. The models developed by [24, 25] have been used to explain how a group of birds or fish manage to move in tight formation as a single entity.

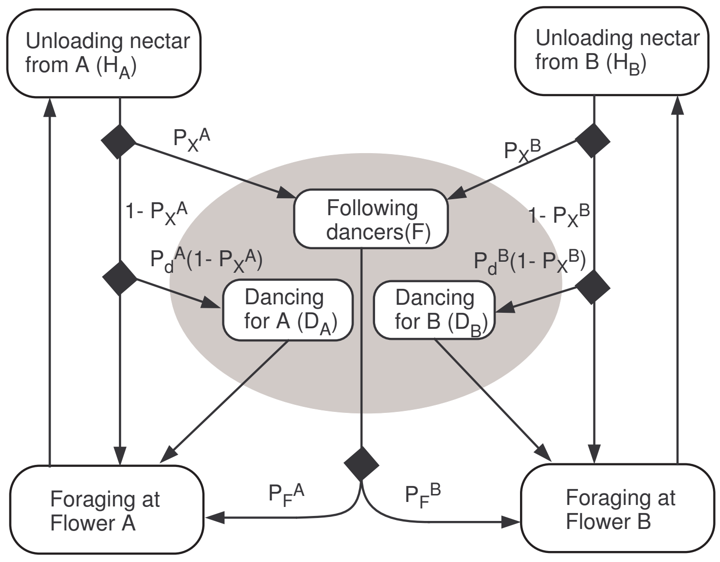


Figure 2.1: The flow diagram illustrates the model of how honey bees allocate the foragers.

Another mathematical model is constructed in [1] to describe the foraging activities of worker honey bees. Each honey bee complies with certain rules to determine where it will go to forage. This process is described by a flow diagram illustrated in Fig. 2.1. At the bifurcations on the diagram, honey bees make decisions on which nectar source to forage and whether to dance - the way honey bees transfer information - or not. The decision-making process is modeled as the probabilities of proceeding various actions. For example, P_X^A represents the probability for one bee to watch other dancers after it unloads the nectar collected from flower A, $P_d^A(1-P_X^A)$ represents the probability of dancing for the flower A and P_F^A represents the probability of following other dancers to forage flower A. Noticing that honey bees make decisions only after receiving limited information from their workmates, [1] proposed a set of simple equations to describe these probabilities, e.g.

$$P_F^A = \frac{D_A d_A}{D_A d_A + D_B d_B}$$

where D_A represent the number of dancers for flower A and d_A is the proportion of time that foragers actually dance. Other probabilities such as P_X^A can be assumed to be a constant. Simulations showed a collective result that was qualitatively similar to what is observed in real bee colonies.

2.2 Models of Ant-Trail Formation

One of the awe-inspiring phenomena in nature is the foraging activity of ant colonies, which includes discovering foods, recruiting nest-mates and forming trails. When an ant finds food, it will recruit other ants around to convey food back to the nest. These co-workers will rapidly form a well-defined trail between the nest and food although they are homogeneously distributed at first. Finding an efficient line between the nest and food seems too complicated a problem for an individual ant to solve, especially if one considers the ant's tiny size relatively to obstacles in the environment, such as stones, stick and crevices. Nonetheless, a colony of ants seem to always be able to complete this task [1]. To explore the intrinsic mechanism that leads to the collective efficiency as opposed to individual clumsiness, several models concerning ant-trail formation have been proposed.

The work of [1] described a model about how ants utilize pheromonal secretions to choose ongoing pathways. According to this model, pheromonal secretions are laid along the paths by ants to keep a trace and recruit other nest-mates. At the same time, pheromonal secretions evaporate as time goes on. When an ant comes to a location where several traces cross, it will try to follow the path with the highest concentration. As illustrated in Fig. 2.2,

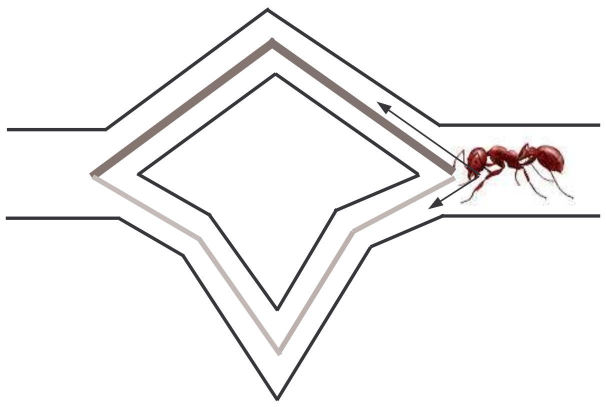


Figure 2.2: An ant chooses the path in accordance with pheromone concentrations

the probability of taking the left branch of a "fork" in the terrain is quantified as

$$P_L = \frac{(k + C_L)^n}{(k + C_L)^n + (k + C_R)^n}$$

The parameters C_L and C_R represent the pheromone concentrations on the left and right branch, n and k are constants corresponding to the degree of nonlinearity of the choice and the attraction of an unmarked branch, respectively. The key point is that the pheromonal secretions play a "positive feedback" role. Although an individual ant knows little about the entire environment and the distribution of its co-workers, simulations show that the colony has the collective potential to find the shortest path.

Another model concerning ant-trail formation on a plane was explored in [26]. The basic rule in this model is that each ant "follows" one of its co-workers instead of measuring pheromonal secretions, as Fig. 2.3 illustrates. In [26], the pheromonal secretions laid by an ant are used to trace its own tail and find its way back to the nest but not to recruit its nest-mates. Paraphrasing [26], the path traveled by a single ant is a curve $x_k(t) : [0, T] \to \mathbb{R}^2$

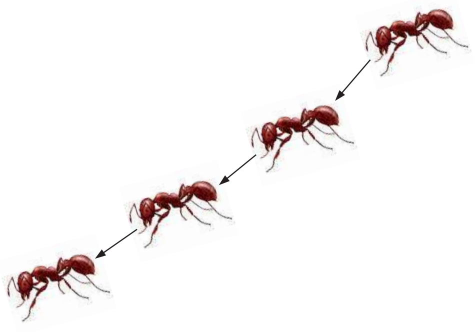


Figure 2.3: Ants find the shortest path joining two members

with $\dot{x}_k = u(x_k)$ ($u \in \mathbb{R}^2$). The boundary conditions for these systems are $x_0(0) = x_0$ and $x_0(T) = x_f$, which represent the starting point (nest) and the target point (food) respectively. Any ant can trace its own trajectory back to the nest so that we have a sequence of ants departing from x_0. Each ant moves with unit speed and there are Δ units of time between the departure time of successive ants. At every instance, each ant except the first one will follow its predecessor by pointing its speed vector in a straight line toward the predecessor. In short, for $k = 1, 2, 3 \ldots$

$$\dot{x}_k(t) = \frac{x_{k-1}(t) - x_k(t)}{\|x_{k-1}(t) - x_k(t)\|}$$

with $x_k(0) = x_0$ for $t \leq k\Delta$ and $x_k(T) = x_f$ if x_k reaches the target x_f. For the case when $x_k(t) \in \mathbb{R}^2$, it has been shown that if the initial ant $x_0(t)$ has access to a sub-optimal path from x_0 to x_f, then the trajectories $\{x_k\}$ will converge to a straight line connecting x_0 and x_f.

2.3 Distributed Covering and Searching

Inspired by the fact that ants and other insects use pheromones for various communication and coordination tasks, [28] developed robust adaptive algorithms to perform tasks requiring the traversal over an unknown region, such as cleaning the floor of an unmapped building. The region to be covered is described by a graph $G = (V, E)$, where every vertex represents an "atomic region" (tile). When agents deployed in the algorithms are traveling on G, they mark the trails by depositing a pheromone, which evaporates as time goes on. By this mechanism, the agents can assign each edge of the graph, which represents the neighborhood relation between two "atoms", with a label of the time that implies the most recent traversal of that edge. An agent visiting vertex $u \in V(G)$ checks the labels on all edges emanating from u, thereafter it goes the direction that was not visited for the longest time by choosing the smallest label. The time t_k needed to cover all edges of the graph by k agents under the "ANT-WALK-1" rule based on the above idea is bounded as

$$t_k \leq n\Delta \left(\rho(G) + \frac{(1+\alpha)n}{k} \right)$$

where Δ is the maximum vertex degree in G, $n = |V(G)|$, α is related to the measurement noise and $\rho(G)$ is the cut-resistant of G. In the same work, the "ANT-WALK-2" rule, a generalization of the famous Depth-First search algorithm, was developed for agents with limited amounts of memory. The time t_k for this rule is bounded as

$$t_k \leq (n\Delta/2) \left\lceil \frac{(1+\alpha)}{k} \right\rceil$$

where the notation is as before.

The work of [29] investigated the performance of cooperative strategies that control autonomous air vehicles searching a dynamic environment to gather information. The proposed framework considers two main components for each agent: distributed learning of the environment and distributed path planning based on the information gathered. The collective results based on a recursive q-step ahead as well as an interleaved planning technique illustrate that the cooperation among vehicles improves the performance. The authors also explored the feasibility of developing coordination control strategies inspired by the social foraging activities of *E. coli*, a common type of bacteria.

2.4 Distributed Localization and Estimation

The study of [30, 31] proposed a method called "Cooperative Positioning System (CPS)" to aleviate the weakness of traditional position identification techniques usually applied in robotics, including dead reckoning and landmark. In that work, a robot group is divided into two teams in order to provide "portable landmarks". At every instance, one team moves and the other stays static, acting as the landmark, then they exchange roles. Therefore, each team can benefit from accurate measurement by utilizing static landmarks, while at the same time, no prior placing of landmarks is required. The drawback is that at least one robot must stay stationary so that the overall speed of the algorithm is restricted.

Another approach has been presented in [32] to simultaneously localize a group of mobile robots with respect to the others' positions. Each robot measures its own motion using its proprioceptive sensors. When two robots x_i, x_j meet, they will share information with one another, then the i^{th} robot will update the estimate of its own position with respect to the j^{th} robot's and the relative distance estimate between the two robots. The proceeding propagation and update are described by the Kalman filter equations in [32]. This method distributes what would be a centralized estimation process among M Kalman filters, each of them operating on a different robot.

2.5 Group Formations

The work of [33] derived a framework that allows robots equipped with range sensors to control their states in order to accomplish the searching or rescuing manipulations. The authors derived three formation controls - "Separation-Bearing Control", "Separation-Separation Control" and "Separation Distance-To-Obstacle Control" - with respect to neighboring robots or obstacles in the environment. The "basic formation" framework is constructed using the above formation controls and is proved to be able to stabilize the formation of a robot team. Lastly, that work outlined a coordination strategy allowing switches between control policies for maintaining the formation in situations with constraints on the sensors, actuators and the environment.

A smooth time-varying feedback control law is developed in [34] to organize formations of multiple nonholonomic wheeled mobile robots. Each robot R_i senses the relative positions of its neighboring robots in its own coordinate system Σ_i. The formation control is described by a vector called "formation vector". Because it is hard to obtain asymptotically stable performance for robots with nonholonomic constraints via smooth static-state feedback controls, the authors utilized a time-varying feedback control law to get the desired velocity for each agent. Using an analytical method based on averaging theory, the group formation under this mechanism is proved to be asymptotically stable.

Another coordinate strategy for vehicle group maneuvers, including translation, rotation, expansion and contraction, is presented in [42] through the construction of artificial potentials and virtual leaders. The control applied on each vehicle is defined as the linear combination of the gradient of these potentials as well as a linear damping term:

$$u_i = -\sum_{j\neq i}^{N} \nabla_{x_i} V_I(x_{ij}) - \sum_{k\neq i}^{M} \nabla_{x_i} V_h(h_{ik}) - K\dot{x}_i$$

where $u_i = \ddot{x}_i$ is the control, x_{ij} is the distance between the i^{th} vehicle and the j^{th} vehicle and h_{ik} is the distance between the i^{th} vehicle and the virtual leader k. The artificial potentials V_I deploy attraction to distant neighbors as well as repulsion for neighbors too close. The accomplishment of desired mission is through controlling the direction of virtual leaders' motion, while the speed of the virtual leaders is to ensure the convergence of the formation. The convergence property is proved by Lyapunov's method.

2.6 Biologically-Motivated Optimization

The work of [36] introduced a search methodology based on the "distributed autocatalytic process" to solve a classical optimization problem - the Traveling Salesman Problem (TSP). Inspired from the fact that ants can use pheromonal secretions to find the shortest path when foraging, [36] utilized an ant team to travel through the towns in TSP. The transition probability from town i to town j for the k^{th} ant is defined as

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{k\in\Omega_k} [\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta} & \text{if } j \in \Omega_k \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where Ω_k is the set of towns reachable by k and $\tau_{ij}(t)$ is the intensity of pheromonal trail on edge (i, j) at time t, which is laid by ants on the edge and evaporates as time goes on. The visibility of the path, η_{ij}, is defined as the reciprocal of the distance between the town i and town j, d_{ij}, i.e. $\eta_{ij} = 1/d_{ij}$. Lastly, α and β are parameters evaluating the relative importance of the trail and the visibility, respectively. Based on Eq. (2.1), [36] developed three algorithms: "ant-cycle","ant density" and "ant-quantity", each based on a slightly different rules by which ants update the $\tau_{ij}(t)$ along their trails. The trajectories of the ant team in each algorithm eventually converges to the optimal tour for the TSP.

The "probabilistic pursuit" algorithm for a group of agents moving on a planar grid was presented in [27]. Briefly, a sequence of agents $A_0, A_1, \ldots$ are moving from the origin at time $t = 0, \Delta, 2\Delta, \ldots$ to a destination. While moving on the grid, A_{n+1} "chases" A_n by making a random choice of a neighboring grid point and moving there. The probability distribution that defines the agent's choice is determined by its relative position to its predecessor, that is

$$A_{n+1}(t+1) = A_{n+1}(t) + \delta_{n+1}(t+1)$$

where $\delta_{n+1}(t+1) \in \{1, -1, j, -j\}$ and

$$\begin{aligned}
\text{Prob}\{\delta_{n+1}(t+1) = \text{sign}(d_x)\} &= \frac{\|d_x\|}{d} \\
\text{Prob}\{\delta_{n+1}(t+1) = j \cdot \text{sign}(d_y)\} &= \frac{\|d_y\|}{d}
\end{aligned}$$

where $A_n(t)$ is the position of the n^{th} agent at time t, $d = \|d_x\| + \|d_y\|$ and d_x, d_y are relative distances between A_n and A_{n+1} at the x and y directions, respectively. Analytical and simulations show that the average trajectories of agents converge to a straight line on the plane. This work is related to the problem of discovering optimal trajectories that will be the focus of this research. It is of course restricted to a discretized plane.

Chapter 3

Biologically Inspired Algorithms for Optimal Control

In this chapter we introduce a class of algorithms inspired by ant-trail formation and discuss their potential advantages. Recall that there are already several effective models of ant-trail formation [1, 26], which explained how a colony of ants find the shortest path length between two points and already seeded some applications [36, 28]. We are particularly interested in the simplicity of the model in [26]. However, [26] only applies to a very narrow domain ($\mathbb{R}^2$ with holonomic, kinematic vehicles). We would like to expand it to a much broader class of optimization problems, including many classical problems in optimal control. Before proceeding with the algorithms, we describe the precise problems we are concerned with.

3.1 Problem Statement and Notation

For our purposes, the agents are assumed to be a number of "copies" of a dynamical system, i.e. for $k = 0, 1, 2 \ldots$

$$\dot{x}_k = f(x_k, u_k) \qquad\qquad x_k(t) \in \mathbb{R}^n, u_k(t) \in \Omega \subset \mathbb{R}^m \tag{3.1}$$

Physically, each copy of Eq. (3.1) could stand for a robot, UAV or other autonomous system.

What we discuss here are some classical trajectory optimization problems for systems evolving under Eq. (3.1) with fixed end points. Each function $x_k(t) : [0, T] \to \mathbb{R}^n$ represents a trajectory defined by the agent's movement. For simplicity, let us start with the problem with fixed final time.

Fixed Final Time Problems

Assume the starting state x_0 and target state x_f are equilibrium points of Eq. (3.1) for $u = 0$[1], i.e.

$$\dot{x}_k(t) = f(x_k(t), 0) = 0 \qquad \text{if } x_k(t) \in \{x_0, x_f\}$$

The problem we are concerned with is finding a trajectory $x^*(t)$ that minimizes the cost function

$$J(x, \dot{x}, t_0, T) = \int_{t_0}^{t_0+T} g(x(t), \dot{x}(t), t)dt \tag{3.2}$$

with $x(t_0) = x_0, x(t_0 + T) = x_f$ and subject to $\dot{x} = f(x, u)$.

The cost function could apply in various categories of optimal control problems, e.g. $g(x(t), \dot{x}(t), t) = \|\dot{x}\|$ (length minimization).

Let $\mathbb{D} \subset \mathbb{R}^n$ be a domain containing states a and b. Assume $0 < \sigma \leq T$ and $t_0 \geq 0$. The optimal trajectory from a to b in fixed T units of time is defined to be $x^*(t)$ ($t \in [t_0, t_0 + T]$) satisfying:

$$J(x^*, \dot{x^*}, t_0, T) = \min_x J(x, \dot{x}, t_0, T) \qquad \text{subject to } x(t_0) = a, x(t_0 + T) = b \tag{3.3}$$

For notational convenience, we define the cost of following $x^*(t)$ for σ units of time as:

$$\eta(a, b, T, t_0, \sigma) \triangleq \int_{t_0}^{t_0+\sigma} g(x^*(t), \dot{x^*}(t), t)dt \qquad \sigma \leq T \tag{3.4}$$

where the optimal trajectory $x^*(t)$ is defined in Eq. (3.3).

For a generic trajectory $x(t)$, we define

$$C(x, t_0, \sigma) \triangleq \int_{t_0}^{t_0+\sigma} g(x(t), \dot{x}(t), t)dt \tag{3.5}$$

to be the cost incurred along $x(t)$ during $[t_0, t_0 + \delta)$.

Free Final Time Problems

Consider a class of optimal control problems with free final time (such as minimum-time control), where we are trying to find a trajectory $x^*(t)$ and a best final time $\Gamma > 0$ that minimize the cost function

$$J_F(x, \dot{x}, t_0) = \int_{t_0}^{t_0+\Gamma} g(x(t), \dot{x}(t), t)dt \tag{3.6}$$

[1]Otherwise we can assume there exist u_0 and u_f such that $f(x_0, u_0) = f(x_f, u_f) = 0$.

with the restriction that $x(t_0) = x_0, x(t_0 + \Gamma) = x_f$ and $\Gamma > 0$. The cost of the optimal trajectory $x^*(t)$ $(t \in [t_0, t_0 + \Gamma])$ from a to b is defined as:

$$J_F(x^*, \dot{x^*}, t_0) \triangleq \min_{x,\Gamma} J_F(x, \dot{x}, t_0) \text{ with } x(t_0) = a, x(t_0 + \Gamma) = b \text{ over all } \Gamma > 0 \quad (3.7)$$

The cost of following the optimal trajectory for σ units of time is defined as

$$\eta_F(a, b, t_0, \sigma) \triangleq \int_{t_0}^{t_0+\sigma} g(x^*(t), \dot{x^*}(t), t)dt \qquad \delta \leq \Gamma \quad (3.8)$$

where $x^*(t)$ is defined in Eq. (3.7).

3.2 A Class of Bio-Inspired Pursuit Algorithms

In the model of ant-trail formation described in [26], each ant is trying to catch up its

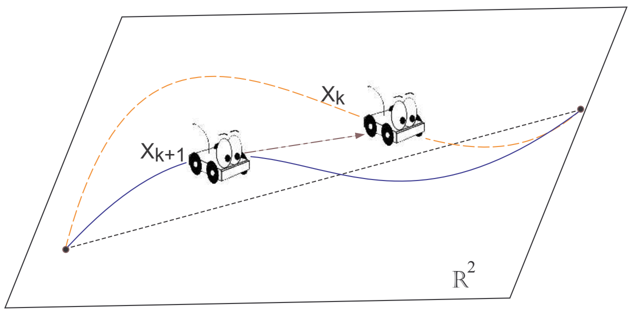


Figure 3.1: A geodesic discovery process on a plane.

predecessor on $\mathbb{R}^2$ in the most "efficient" way, namely by pointing its velocity vector towards its predecessor. The trajectories generated by the movements of ants are gradually optimized and the trajectory sequence converges to a straight line on $\mathbb{R}^2$, as illustrated in Fig. 3.1. The work in [18] expanded the above approach to uneven terrains. Both [18] and [26] separated the task of finding a geodesic over long distances into many simpler tasks of seeking geodesics connecting nearby points. The difficulty of "following" increases in accordance with the distance between the predecessor and the successor and with the complexity of the terrain. It is easer for an ant to aim at its leader on $\mathbb{R}^2$ and move on a shortest path toward it if they are closer, whether on a plane or on a terrain. Same is for a robot that havs limited sensing range and computing ability.

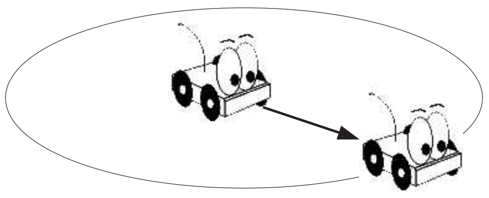


Figure 3.2: It is easier to solve an optimization problem within a “small” region.

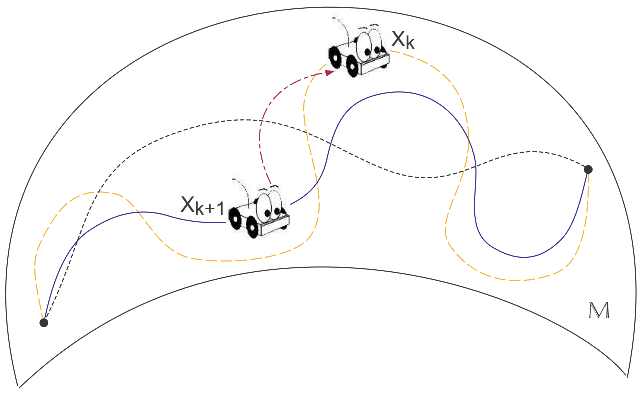


Figure 3.3: Expanding the algorithm to more general optimization problems on a manifold.

We are interested in generalizing the existing approaches in [26, 18] to a much broader class of optimization problems in Eq. (3.3), (3.6), using an iterative strategy requiring little communication as well as short-range sensing. There is an analogy here between optimal control problems that are easier to solve where the boundary conditions are “close” to one another, and members of a collective that are easier to follow from a close distance, as Fig. 3.2 illustrates. Our idea is to seek optimal trajectories locally, by means of “*local pursuit*”, and combine the efforts of a group of agents to gradually optimize an initial solution.

Our approach will be to propose a set of iterating rules that somehow generalize the idea of pursuit to settings with non-trivial geometry, and agents with non-trivial dynamics. If this approach succeeds, then complicated tasks could be separated into simpler tasks and accomplished by a group of “inexpensive” agents. The following is an algorithm that prescribes the evolving of a group, given an initial feasible trajectory.

Algorithm 1 (Sampled Local Pursuit): *Identify two states* x_0 *and* x_f *on* $\mathbb{D}$. *Let*

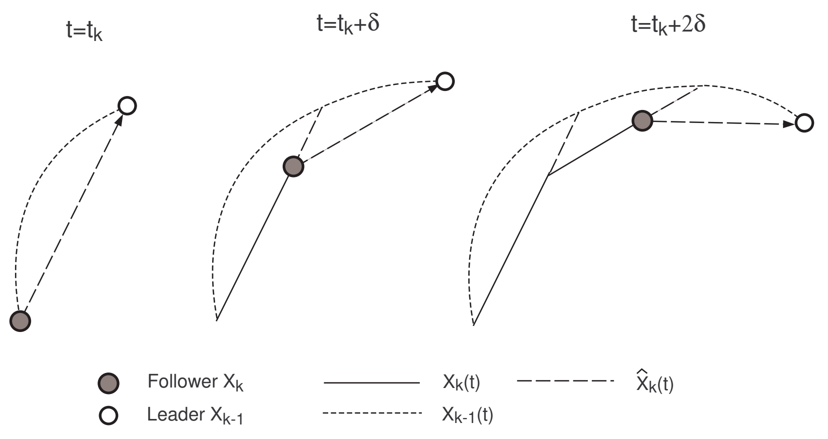


Figure 3.4: A snapshot of the updating processes executed by the k^{th} agent.

$x_0(t)$ $(t \in [0,T])$ be an initial trajectory satisfying Eq. (3.1) with $x_0(0) = x_0, x_0(T) = x_f$. Choose the following interval Δ and updating interval δ such that $0 < \delta < \Delta \leq T$. Then follow the next rules for the k^{th} agent.

1. *For $k = 1, 2, 3 \ldots$, let the $t_k = k\Delta$ be the starting time of the k^{th} agent. Let $u_k(t) = 0, x_k(t) = x_0$ for $0 \leq t \leq t_k$.*

2. *When $t = t_k + i\delta, i = 0, 1, 2, 3, \ldots$, calculate $u_t^*(\tau)$ such that $f(x_k(\tau), u_t^*(\tau)) = \dot{x}_k^*(\tau)$, where*
$$x_k^*(\tau) \text{ achieves } \begin{cases} \eta(x_k(t), x_{k-1}(t), \Delta, t, \Delta), & \tau \in [t, t+\Delta] \quad \text{if } \Delta + i\delta < T \\ \eta(x_k(t), x_f, t_k + T - t, t, t_k + T - t), & \tau \in [t, t_k + T] \quad \text{otherwise} \end{cases}$$

3. *Apply $u_k(t) = u_{t_k+i\delta}^*(t - t_k - i\delta)$ to the k^{th} agent for $t \in [t_k + i\delta, t_k + (i+1)\delta)$ if $\Delta + i\delta < T$ or $t \in [t_k + i\delta, t_k + T)$ otherwise.*

Repeat from step 2, until the k^{th} agent reaches x_f.

This is a "sampled" version local pursuit because agents are only required to update their trajectories a finite number of times. There are two adjustable parameters: the "following interval" Δ and the "updating interval" δ. Usually we take $0 \leq \delta < \Delta$. We will refer to the times $t_k^i = t_k + i\delta, i = 0, 1, 2, 3 \ldots$ as the "updating times". Notice that the SLP algorithm yields a well-defined trajectory $x_k(t)$ on $[0,T]$, if given $x_{k-1}(t)$. The resulting trajectory is continuous but not necessarily smooth at the time interval $[t_k, t_k + T]$. A snapshot of the iteratively updating processes is illustrated in Fig. 3.4.

According to the SLP algorithm, agents leave the starting state x_0 one after another, each in Δ units of time after its predecessor. That is, if the $(k-1)^{th}$ agent leaves the starting state at time t_{k-1}, the k^{th} agent will leave it at $t_k = t_{k-1} + \Delta$. We assume the number of agents in the group is large and label each agent by an integer k so that we can utilize $x_k(t)$ to denote the k^{th} agent's trajectory[2]. Each agent moves to pursue its predecessor. If we denote the $(k-1)^{th}$ agent as the "leader" during this pursuit relationship, the k^{th} agent will be denoted as the "follower". At each $t = t_k^i$, the follower calculates the optimal control $u_t^*(\tau)$ $(\tau \in [t, t+\Delta))$ that steers it from $x_k(t)$ to $x_{k-1}(t)$ over Δ units of time, i.e. from its current state to the leader's current state. Then during $[t_k + i\delta, t_k + (i+1)\delta]$, the follower moves along the trajectory driven by $u_t^*(\tau)$, and the process repeats until the follower reaches x_f.

For notational convenience, we define the *planned* trajectories, denoted by $\hat{x}(t)$, to be the trajectories along which the follower plans to move at $t_k + i\delta$ but may not do so because it will update its future trajectory at $t_k + (i+1)\delta$. In other words, the planned trajectories are the trajectory driven by $u^*_{t_k+i\delta}(\tau)$ for the time period of $[t_k + (i+1)\delta, t_k + i\delta + \Delta]$, while it may not actually be executed because the next updating result, $u^*_{t_k+(i+1)\delta}(\tau)$, will drive the agent to move along different trajectories. The *realized* trajectories, denoted by $x(t)$, are defined as the trajectories along which the follower actually moves. Referring to Fig. 3.4, the planned trajectories and realized trajectories are represented by the dashed lines and solid lines, respectively.

If we let $\delta \to 0$ in SLP, we will obtain the following continuous local pursuit algorithm:

Algorithm 2 (Continuous Local Pursuit): *Identify two states x_0 and x_f on $\mathbb{D}$. Let $x_0(t)$ $(t \in [0,T])$ be an initial trajectory satisfying Eq. (3.1) with $x_0(0) = x_0, x_0(T) = x_f$. Choose the following interval Δ such that $0 < \Delta \le T$. Then follow the next rules for the k^{th} agent.*

1. *For $k = 1, 2, 3 \ldots$, let $t_k = k\Delta$ be the starting time of k^{th} agent. Let $u_k(t) = 0, x_k(t) = x_0$ for $0 \le t \le t_k$.*
2. *Calculate $u_t^*(\tau)$ for all $t \in [t_k, t_k + T]$ such that $f(x_k(\tau), u_t^*(\tau)) = \dot{x}_k^*(\tau)$, where*
$$x_k^*(\tau) \text{ achieves} \begin{cases} \eta(x_k(t), x_{k-1}(t), \Delta, t, \Delta), & \tau \in [t, t+\Delta] \quad \text{if } t < t_k + T - \Delta \\ \eta(x_k(t), x_f, t_k + T - t, t, t_k + T - t), & \tau \in [t, t_k + T] \quad \text{otherwise} \end{cases}$$
3. *Apply $u_k(t) = u_t^*(0)$ to the k^{th} agent.*

Repeat from step 2, until the k^{th} agent reaches x_f.

Due to the limitations of each agent's computing capability, it might be more expedient to apply the sampled local pursuit (SLP) because the agents only need to update their

[2]From now on, we will utilize $x_k(t)$ to denote both the k^{th} agent and its trajectory.

trajectories finite times instead of continuously in CLP. However CLP does not require storage of calculated results so it is more favored in situations where the update is easily to be carried out.

If we are dealing with a free final time optimization problem, then the SLP and CLP algorithms must be altered so that agents optimized their trajectories connecting them pairwise with respect to both u and the final time.

Continuous local pursuit is thus altered as follows[3].

Algorithm 3 (Free Final Time Local Pursuit): *In Algorithm 2 replace the step 2 with:*
2.' Calculate $u_t^*(\tau)$ for all $t \in [t_k, t_k + T]$ such that $f(x_k(\tau), u_t^*(\tau)) = \dot{x}_k^*(\tau)$, and $x_k^*(\tau)$ achieves $\eta_F(x_k(t), x_{k-1}(t), t, \Gamma)$ $(\tau \in [t, t + \Gamma])$, where η_F is given by Eq. (3.8).

3.3 Algorithm Advantages

Each agent that participates in local pursuit is only required to calculate the optimal trajectory from itself to its nearby leader. Meanwhile the "distance" between them can be limited by selecting an appropriate following interval Δ. Therefore every agent only needs to sense the environment within a limited region when proceeding pursuit processes. This is preferable to obtaining a global map via random exploration with limited sensor range. For example, it would be difficult and wasteful for a single robot to obtain the entire map of an unknown terrain. Even if a group of agents can be dispersed and each composes a map "patch" around itself, it is not guaranteed that the composition of these patches covers the whole environment, or at least covers the region containing the optimal trajectory.

Even if enough patches covering the entire environment have been collected, the fusion of a composite map still requires a large amount of information communication. A powerful agent is also needed to stitch the scattered maps using sophisticated fusion algorithms. This means at least one agent in the group has enough memory, communication bandwidth and computing ability to dealing with the collecting and fusing tasks concerning the entire environment. In contrast, in local pursuit, there is no requirement for agents to exchange local maps that they sense. Agents only have to communicate in very limited ways, by using vision to track one another or by communicating in primitive ways to signal their locations, e.g. sound or radio emission.

Furthermore, even if an effective map could be obtained, solving optimization problems

[3]In SLP, it can not be guaranteed that at every updating time the minimum time to reach the leader Γ is greater than or equal to the updating interval δ. If $\Gamma < \delta$, then extra costs might be incurred. Based on the above consideration, we only develop the Free Final Time Local Pursuit (FFTLP) in continuous version so that $\delta < \Gamma$ is guaranteed.

over an sophisticated environment, especially in an environment containing different kinds of coordinate patches, requires large amounts of calculations. The example includes finding geodesics on a terrain with mountains and basins. The most often used technique in such situations is numerical method. As we shall see later, using numerical method over long distances may leads to huge amounts of calculations. However, local pursuit only requires computing optima within small regions so that fewer calculations are needed.

In summary, local pursuit introduces a way to obtain the locally optimal trajectory[4] over distance by many short pieces generated via an ordered sequence of identical agents, meanwhile it only requires local knowledge about the environment as well as calculation of optimal trajectories within small regions. Thus, a complicated optimization problem could be solved by a group of cost-effective agents. The trade-off is that each agent must compute locally optimal trajectories more than once. However, the deployment of a group of cheap agents using local pursuit does show various advantages with cost and reliability consideration, if compared with achieving the same task by a single, expensive agent.

[4]This conclusion will be proved in next chapter.

Chapter 4

Current Progress

In this chapter we will investigate the collective behavior of the group involved under local pursuit. Recall that each algorithm defines an ordered sequence of trajectories $\{x_k(t)\}$. The convergence of the sequence involved under SLP will be first explored. Then the limiting trajectory will be proved to be locally optimal, this is exactly the collective property we are seeking to obtain. Similar results will be explored in CLP and FFTLP. Special cases concerning path length and time minimizing problems will be introduced because of their prevalence in practice. Lastly, simulation experiments are provided to illustrate our results.

4.1 Results on Sampled Local Pursuit

We would like to investigate the property of the limiting trajectory generated by the group, i.e. $x_k(t)$ as $k \to \infty$. The convergence of the trajectories' cost will be explored first, then the convergence of trajectories themselves, $\{x_k(t)\}$. After that, we show that the limiting trajectory of the sequence, denoted as $x_\infty(t)$, is locally optimal.

Lemma 4.1 (Convergence of Cost in SLP): *Assume a group of agents $x_0, x_1, \ldots, x_k$ evolve under "Sampled Local Pursuit" with starting state x_0 and target state x_f. Suppose an initial control/trajectory pair, $\{u_0(t), x_0(t)\}$ $(t \in [0,T])$, satisfying $x_0(t) = x_0$ and $x_0(T) = x_f$ is given. If the updating time satisfies $0 < \delta \leq \Delta$, then the cost of the iterated trajectories will converge, i.e. $\lim_{k\to\infty} C(x_k, t_k, T)$ exists.*
Sketch of Proof: Given an existing optimal control problem, the cost of any trajectory satisfying the boundary conditions is bounded below. By investigating the pursuit process between $x_k(t)$ and $x_{k-1}(t)$ pairwise, we can prove that $C(x_k, t_k, T) \leq C(x_{k-1}, t_{k-1}, T)$. This is enough to show the convergence of the sequence $\{C(x_k, t_k, T)\}$. See Appendix A.2 for the detailed proof. $\square$

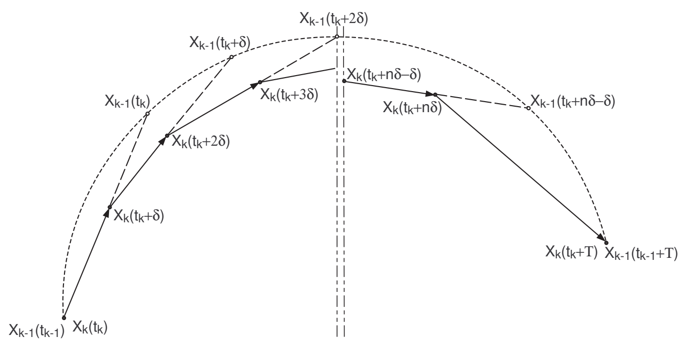


Figure 4.1: Sketch of the pursuit process pairwise in SLP

Nonetheless, the convergence of trajectories' cost does not imply the convergence of the trajectories themselves. If there exist multiple locally optimal trajectories connecting the leader and follower at the updating times, then the convergence of trajectories is not guaranteed, i.e. Lemma 4.1 defines an equivalence class of trajectories with the same cost.

If we restrict the pursuit process to take place within a "small" region by selecting Δ sufficiently small, e.g. agents follow close to one another, there will exist a unique locally optimal trajectory from the follower to the leader at every updating time $t_k + i\delta$. Thereafter we obtain the following result:

Lemma 4.2 (Uniqueness of the Limiting Trajectory): *If at each updating time, the locally optimal trajectory obtained through SLP is unique, then the limiting trajectory $x_\infty(t)$ is also unique.*
Sketch of Proof: We will show that if there exist more than one trajectories that $x_k(t)$ might take, for k large enough, then the cost of one trajectory must be less than the others. This contradicts to what we have obtained from Lemma 4.1, which shows that the limiting trajectories should have the same cost if they exist. See Appendix A.3 for the details of the proof.
$\square$

The locally optimal trajectories obtained at every updating time are smooth in many optimal control problems, e.g. the solution to the Euler-Lagrange equation in calculus of variations. Nonetheless, $x_k(t)$ is only known to be piecewise smooth. For example, in $\mathbb{R}^2$ with $\dot{x}_k = u_k$, if the locally optimal trajectories are straight lines, $x_k(t)$ is not smooth for there exists a corner at the joint of two segments. However, we can show that the limiting trajectory is smooth in the time interval $[0, T]$, the locally optimal trajectories obtained at every updating time are smooth. The following definitions will be necessary for discussing the properties of the limiting trajectory.

Definition 4.1: *Let $\gamma_1(t)$ and $\gamma_2(t)$ be trajectories of Eq. (3.1), defined on a time interval I_1 and another time interval I_2 respectively, where $I_1 \cap I_2 \neq \emptyset$. We say that γ_1 and γ_2* ***overlap*** *if $\gamma_1(t) = \gamma_2(t)$ for all $t \in I_1 \cap I_2$.*

Definition 4.2: *Let $\gamma_1(t)$ and $\gamma_2(t)$ be trajectories of Eq. (3.1), defined on a time interval I_1 and another time interval I_2 respectively, where $I_1 \cap I_2 \neq \emptyset$. The* ***composition*** *of $\gamma_1(t)$ and $\gamma_2(t)$ on the interval $I_1 \cup I_2$ is defined as*

$$\gamma_1 \circ \gamma_2 \triangleq \begin{cases} \gamma_1(t) & t \in I_1, t \notin I_2 - I_1 \cap I_2 \\ \gamma_2(t) & t \notin I_1, t \in I_2 - I_1 \cap I_2 \end{cases}$$

Lemma 4.3 (Smoothness of Composition): *Suppose that in Lemma 4.1 the updating interval δ and the following interval Δ satisfy that $0 < \delta < \Delta$, then the planned trajectory $\hat{x}(t)$ and realized trajectory $x(t)$ of the limiting trajectory overlap. Furthermore, if the locally optimal trajectories obtained at every updating time are smooth, then the limiting trajectory is also smooth.*

Sketch of Proof: We will first explore that the planned trajectory and realized trajectory of $x_\infty(t)$ overlap by contradiction. Then it is shown that the limiting trajectory is piecewise smooth and its neighboring segments overlap, the smoothness of the limiting trajectory over the entire time interval is an immediate consequence. See Appendix A.4 for the details of the proof. $\square$

Before proceeding to the main theorem, we are required to define the following condition.

Condition 4.1: *Assume there exists an $\varepsilon > 0$ such that for all $a, b_1, b_2 \in \mathbb{D}$ and all $\Delta > 0$, the optimal cost $\eta(a, b_1, \Delta, 0, \Delta)$ from a to b_1 and $\eta(a, b_2, \Delta, 0, \Delta)$ from a to b_2 satisfy*

$$\|b_1 - b_2\| < \varepsilon \Rightarrow \|\eta(a, b_1, \Delta, 0, \Delta) - \eta(a, b_2, \Delta, 0, \Delta)\| < \mathcal{L}\Delta \tag{4.1}$$

for some constants $\mathcal{L}$ independent of Δ.

A piecewise locally optimal trajectory is not necessarily optimal. However, the composition of overlapping locally optimal trajectories is locally optimal if Condition 4.1 is satisfied.

Lemma 4.4 (Composition of Optimal Trajectories): *Let $\gamma_1(t)$ and $\gamma_2(t)$ be overlapped locally optimal trajectories defined on a time interval I_1 and another time interval I_2 respectively, where $I_1 \cap\ I_2 \neq \emptyset$. If Condition 4.1 is satisfied, then the composition $\gamma_1 \circ \gamma_2$ is locally optimal on $I_1 \cup I_2$.*

Sketch of Proof: Suppose that the composition (call it $x^*(t)$) is not locally optimal, then there must exist another trajectory (call it $x(t)$) nearby such that $\|x(t) - x^*(t)\|_\infty < \varepsilon$ and $C(x(t), 0, T) < C(x^*(t), 0, T)$. We can then use Condition 4.1 to obtain a contradiction, namely that $C(x(t), 0, T) > C(x^*(t), 0, T)$. See Appendix A.5 for the complete proof. $\square$

The next theorem is an immediate consequence of the above lemmas.

Theorem 4.1 (Sampled Local Pursuit): *Suppose a group of agents $\{x_k\}$ evolve under sampled local pursuit and at each updating time $t = t_k + i\delta$, the locally optimal trajectory from $x_k(t)$ to $x_{k-1}(t)$ is unique. If the updating interval and following interval satisfy $0 < \delta < \Delta$ and Condition 4.1 is satisfied, then the trajectory sequence converges to a unique local optimum. Furthermore, if the locally optimal trajectories at every updating time are smooth, the limiting trajectory is also smooth.*

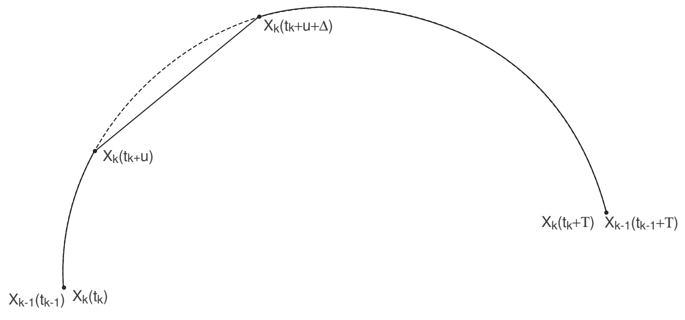
Proof: From Lemma 4.2, the limiting trajectory is unique. We know that $x_\infty(t)$ ($t \in [0, \Delta)$) and $x_\infty(t)$ ($t \in [\delta, \delta + \Delta)$) are locally optimal for the realized trajectory and planned trajectories overlap (Lemma 4.3). The optimality of $x_\infty(t)$ ($t \in [0, \delta + \Delta)$) follows from Lemma 4.4. Repeating this argument on $[i\delta, i\delta + \Delta]$ ($i = 0, 1, 2 \ldots$) leads to the result that $x_\infty(t)$ ($t \in [0, T]$) is locally optimal. The proof of smoothness follows from a similar argument. $\square$

4.2 Results on Continuous Local Pursuit

In the case of continuous local pursuit, the follower keeps on updating its movement at every $t \in [t_k, t_k + T]$, i.e. the updating interval $\delta \to 0$. Similar to the sampled local pursuit, we assume the selection of Δ guarantees that at every updating time there is a unique optimal trajectory from the follower to the leader. We will first show that a single update to the leader's trajectory will result in less cost than what is incurred by the leader, no matter when the update occurs. Then we will explore the convergence of the trajectories' cost in CLP. The remaining arguments are quiet similar to what we had discussed in SLP.

Lemma 4.5: *Let $\lambda \in [0, T)$. Suppose that a follower replicates the leader's trajectory on $t \in [t_k, t_k + \lambda) \cup [t_k + \lambda + \Delta, t_k + T]$ if $\lambda \leq T - \Delta$ (or $t \in [t_k + \lambda, t_k + T]$ if $\lambda > T - \Delta$), while*

during $[t_k + \lambda, t_k + \lambda + \Delta]$ it follows the optimal trajectory joining $x_k(t_k + \lambda)$ and $x_{k-1}(t_k + \lambda)$ in Δ (or $(T - \lambda)$) units of time. Then the cost along the follower's trajectory will be no greater than the leader's.
Sketch of Proof: We can investigate the cost along the follower and the leader, respectively. The overlapping parts of the leader's and follower's trajectories will lead to equal



$X_k(t_k+u+\Delta)$
$X_k(t_{k+u})$
$X_k(t_k+T)$ $X_{k-1}(t_{k-1}+T)$
$X_{k-1}(t_{k-1})$ $X_k(t_k)$

Figure 4.2: Sketch of a single update.

costs, while the follower incurs less cost during $[t_k + \lambda, t_k + \lambda + \Delta]$. It follows that the whole cost along the follower is less than the leader's. See Appendix A.6 for the complete proof. □

Lemma 4.6 (Convergence of Cost in CLP): *In the case of continuous local pursuit, the cost of the iterated trajectories converges.*
Sketch of Proof: The movement of the k^{th} agent under CLP can be interpreted as the consequence of applying infinitely moving "updates" to the leader's trajectory. From Lemma 4.5, each update leads to non-increasing cost so that infinite times of update will also lead to less or equal cost for the follower. See Appendix A.7 for the details of the proof. □

Now the main result concerning continuous local pursuit can be derived easily by an argument similar to what was used for sampled local pursuit.

Theorem 4.2 (Continuous Local Pursuit): *Suppose a group of agents evolve under continuous local pursuit and that at every updating time t, the locally optimal trajectory from $x_k(t)$ to $x_{k-1}(t)$ is unique. Then the limiting trajectory obtained is unique and locally optimal. It is smooth also if the locally optimal trajectories calculated at every updating time are smooth.*
Proof: First, we assume there are two different limiting trajectories $x_1(t)$ and $x_2(t)$. The proof of Lemma 4.2 shows that if there exist updates during the non-overlapping parts of successive trajectories $x_1(t)$ and $x_2(t)$, the whole cost along the follower will be less than the

leader's, even in the case where infinite updates occur because the number of updates does not change this property. If there exist more than one limiting trajectories, the decrease of cost from the leader to the follower contradicts the fact that all the limiting trajectories must have the same cost. Therefore the limiting trajectory is unique. It follows that $x_{k-1}(t-\Delta) = x_k(t)$ if $x_{k-1}(t) = x_\infty(t - t_{k-1})$. If we pick a δ_1 such that $0 < \delta_1 < \Delta$, the limiting trajectory is piecewise smooth and locally optimal. Using the arguments in Lemma 4.3 and Lemma 4.4 we can say $x_\infty(t)$ is smooth and locally optimal over the entire time interval. □

4.3 Results on Free Final Time Local Pursuit

Notice that Lemma 4.5 still holds for the free final time local pursuit. The convergence of the trajectories' cost is easily to obtain using the similar arguments in Lemma 4.6. Using the similar argument in Theorem 4.2 and changing the argument to free final time version will yield the following result.

Theorem 4.3 (Free End-Time Local Pursuit): *Suppose a group of agents evolve under free final time local pursuit and at every updating time t, the locally optimal trajectory from $x_k(t)$ to $x_{k-1}(t)$ with free final time is unique. Then the limiting trajectory is unique and locally optimal, it is also smooth if the locally optimal trajectories calculated at every updating time are smooth.*

Proof: The proof is simple and will be omitted here. □

4.4 Summary

Until now, we have seen that each algorithm (SLP, CLP, FFTLP) will generate an interesting "collective pattern" - the local optimum for proposed optimal control problem. Although each agent only solves the optimal control problem within a small region (limited by Δ), the trajectories generated by them are gradually optimized. Each agent "learns" from its predecessor and the limiting trajectory exhibits the collective intellect of the group. Therefore, a complicated task (optimizing over long distance) is separated into small tasks requiring less capabilities of sensing, communicating and computing.

Our algorithms fall into the category of "learning by repetition". Newton's method and gradient methods are well-known examples in this category, and are usually applied to solve extremal problems in finite dimensional vector spaces [6]. Extensions of such methods in function spaces also enable the development of trajectory optimization algorithms through repetition. For example, the work of [40] utilized a developed gradient method to iteratively optimize the control for a specified dynamic system . The control $u(t)$ is derived by

$$\frac{du}{dt} = -\alpha \frac{\partial}{\partial u} \left[\frac{\partial W(x,t)}{\partial x} X(x,u) \right] \tag{4.2}$$

where $X(x,u) = \dot{x}(t)$ are the system dynamics and $W(x,t)$ is the minimal cost of reaching the final state x_f provided with the initial state is $x(t_0) = x$. Eq. (4.2) converges to the optimal control $u^*(t)$ and $x^*(t)$ if the optimal control is smooth.

However, existing algorithms usually require the cost function and the control to be partial differentiable. To proceed with the above algorithm, they also need to store and describe the entire x_k, in order to get x_{k+1}. Moreover, to obtain a smooth curve, infinitely small time increments are required so that laborious calculations are introduced. All these factors hinder the application of these algorithms in decentralized systems whose members are working cooperatively.

In contrast, our proposed algorithms are suitable for a large class of optimization problems and do not suffer from the above drawbacks. For example, our algorithms could be applied in the situations where the control and trajectory are not smooth such as Bang-bang control. The computing requirement for each agent could be limited by defining an appropriate Δ. Furthermore, each agent only need very limited information of its predecessor so that multiple agents could work together to achieve the most effectiveness.

4.5 Special Cases: Length and Time Minimization

We have the additional interesting results for the trajectory optimization problems that often involve reaching a desired target state with minimum path length or end time. We state it as follows.

Theorem 4.3 : *If the time rate of the change of the cost along a trajectory is independent on $x_k(t)$ for all t, then the minimum cost from the follower to the leader with free final time is strictly decreasing under local pursuit, unless the leader moves along a locally optimal trajectory.*

Proof: Let $\rho(a,b) = J_F(x^*, \dot{x}^*, \tau)$ be the minimum cost to steer system from state a to another b. For the pursuit process shown in Fig. 4.3, We have that

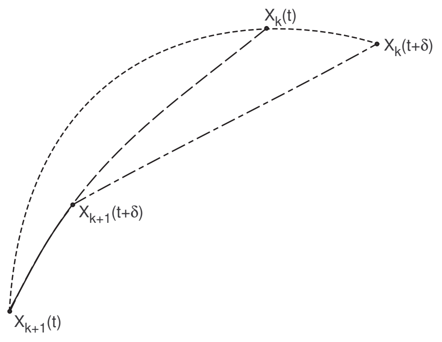


Figure 4.3: The minimum cost from $x_{k+1}(t)$ to $x_k(t)$ is decreasing if dC/dt is independent.

$$\begin{aligned}
\rho(x_{k+1}(t+\delta), x_k(t+\delta)) &\leq \rho(x_{k+1}(t+\delta), x_k(t)) + \rho(x_k(t), x_k(t+\delta)) \\
&\leq \rho(x_{k+1}(t+\delta), x_k(t)) + C(x_k(t), t, \delta) \\
&= \rho(x_{k+1}(t+\delta), x_k(t)) + C(x_{k+1}(t), t, \delta) \\
&= \rho(x_{k+1}(t), x_k(t)) \qquad (4.3)
\end{aligned}$$

If the equalities hold in Eq. (4.3) then $x_k(t)$ must be moving along an optimal trajectory. $\square$

This result has a variety of applications, e.g. the minimum time control problem

$$J(x, \dot{x}, 0, T) = T \qquad \|\ddot{x}\| \leq 1 \qquad (4.4)$$

whose solution could be obtained via the maximum principle; or the minimum path length problem with the condition that all agents are moving on unit speed

$$J(x, \dot{x}, 0, T) = T \qquad \text{with } \|\dot{x}\| = 1, T \text{ is free} \qquad (4.5)$$

4.6 Simulations

We now present some simulation results concerning application of local pursuit in different optimal control problems.

Sampled Local Pursuit

To illustrate the effectiveness of sampled local pursuit, we solve the minimum path length problem on $\mathbb{R}^2$ with boundary conditions $x(0) = 0, x(1) = 1$. Obviously the optimal trajectory is a straight line. We set $\delta = 0.25, \Delta = 0.5, T = 1$. Fig. 4.4 shows 5 trajectories iterated from sampled local pursuit. The 5^{th} trajectory is close to straight line.

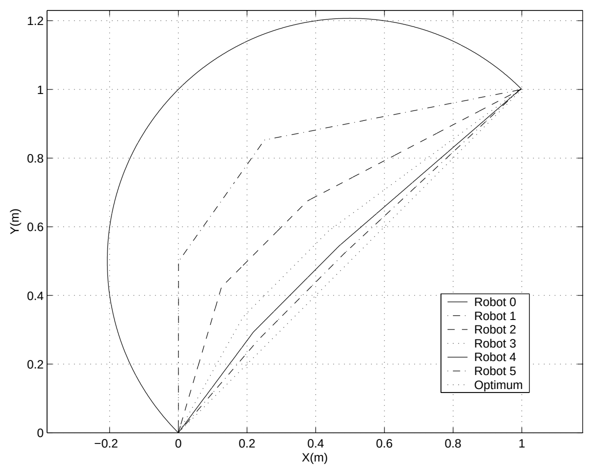


Figure 4.4: Iterated trajectories of minimum length problem through SLP on $\mathbb{R}^2$

A Lagrangian Example

Fig. 4.5 illustrates the application of continuous local pursuit in systems with drift. Here

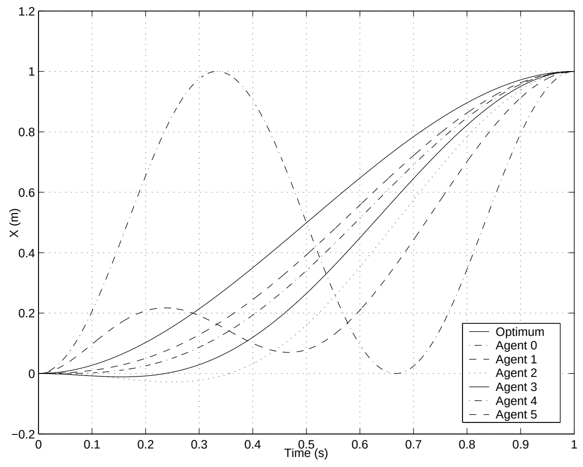


Figure 4.5: Iterated trajectories for the Lagrangian problem through CLP with $\Delta = 0.5$

the system dynamic is

$$\ddot{x}(t) + x(t) = u(t)$$

and we want to minimize

$$\int_0^1 (\dot{x}(t)^2 + u(t)^2)dt \qquad \text{with } x(0) = 0, x(1) = 1$$

The locally optimal trajectory could be obtained through Euler-Lagrange equation from calculus of variations. The following interval Δ is set to be 0.5. Fig. 4.5 shows that the trajectory sequence converges to the optimum.

Minimum Time Control

Consider the following second-order system

$$\ddot{x} = u \qquad \|u\| \leq 1$$

And we want to minimize the cost $J(x, \dot{x}, 0, T) = T$ with the boundary conditions of $x(0) = \pi, x(T) = 0$. From maximum principle it is well known that the optimal control for this problem is the Bang-bang control.

$$u^*(t) = \begin{cases} -1 & \text{if } t \in [0, T/2) \\ 1 & \text{if } t \in [T/2, T] \end{cases} \tag{4.6}$$

With the following interval $\Delta = 0.3\pi$, as Fig. 4.6 illustrates, the trajectory of 6th agent is essentially under optimal control, which means the convergence is really rapid.

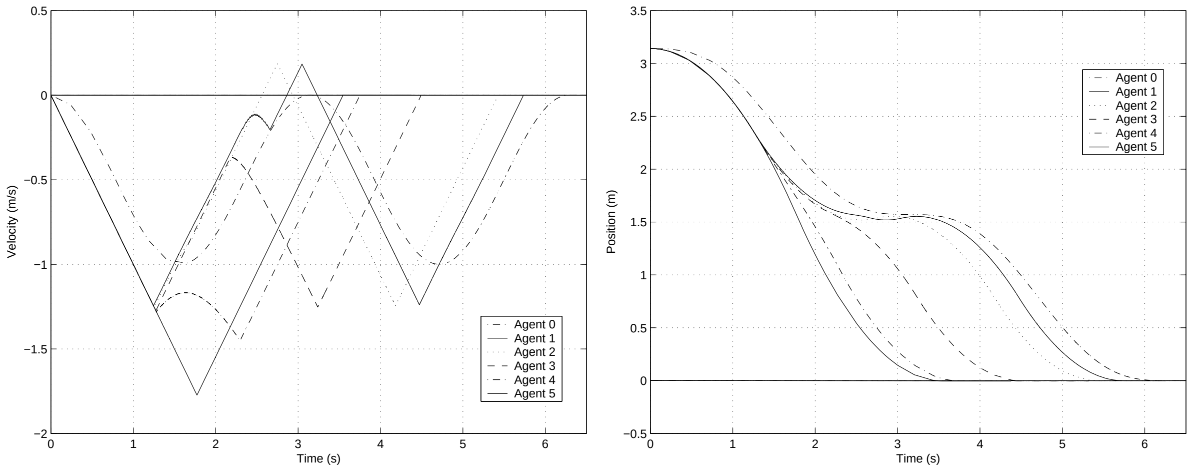


Figure 4.6: Iterated trajectories for minimum time control problem through FFTLP with $\Delta = 0.3\pi$

Geodesic Discovery

Now we will show a geodesic discovery example that involves complicated calculation over entire environment but relatively simpler in local patches. This example simulates two hills by two cones. The starting state is $(3500, 0, 0)$ and the end state is $(-1300, 0, 0)$. The first

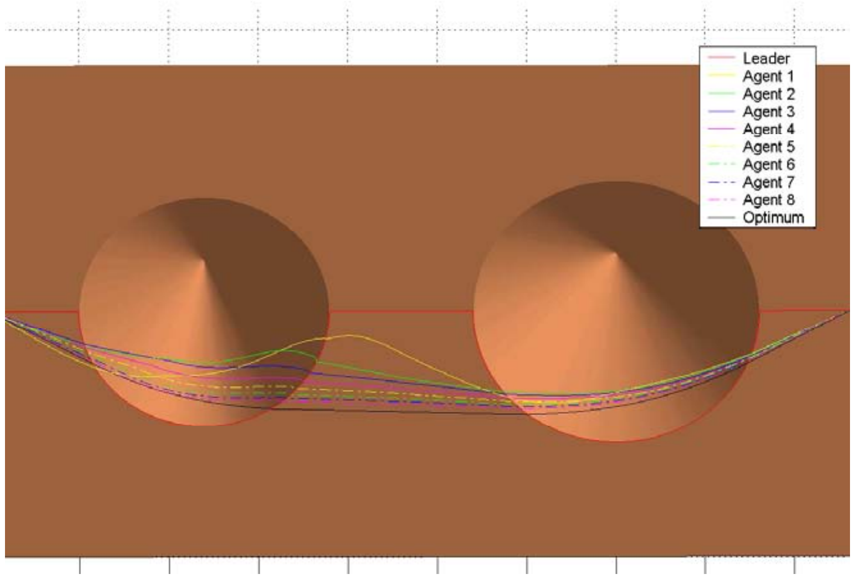


Figure 4.7: Iterated trajectories for the geodesic discovery problem through CLP.

agent moves on a trajectory that follows along the border of the cones. The geodesic over large distance is not easy to compute because not only it demands knowledge over the entire map but also there are 4 coordinate switches along the path[1].

However, if we set $\Delta = 0.2T$, the follower is at most required to do calculation with one coordinate switch so that the amount of calculation at every step is decreased, compared to computing over the whole map. As Fig. 4.7 illustrates, the iterated trajectories converge to the optimum.

[1]If applied numerical method over the entire map, the number of the time segments is 4.

Chapter 5

Ongoing Work

In this chapter we will discuss some ongoing research directions related to local pursuit, as listed as follows.

- Notice that a large category of optimal control problems are which involve free final state or "point-to-set" problems as oppose to point boundary conditions problems. We would like to generalize our pursuit algorithm to such problems.
- The limiting trajectory obtained from local pursuit may converge to a global optimum or a local optimum, depending on the parameters of the algorithm. We will explore that dependence and determine which optimum the trajectory sequence converges to.
- The performance of local pursuit with noisy measurements will be considered due to its relevance in practice. We want to know whether the agents can estimate the solution in the absence of precise sensor readings.
- We will explore the advantages of local pursuit in the numerical computation of optimal trajectories.

Finally, we will look into the development of other biologically inspired algorithms for complicated tasks in engineering or other fields. The potential tasks and ongoing steps will be outline next.

5.1 Optimal Control Problems with Free Final State

Many optimal control problems with fixed final time include a penalty to the final state but do not impose any constraints on it, i.e.

$$J(x, \dot{x}, t_0, T) = Q(x(t_0 + T)) + \int_{t_0}^{t_0+T} g(x(t), \dot{x}(t), t)dt \tag{5.1}$$

The example includes the LQR problems, which could be solved by introducing a feedback control and a Riccati equation, as we know.

We would like to modify local pursuit to apply to this class of problems, if possible. Recall that in local pursuit we are gradually optimizing our initial solution, it seems that if the cost incurred by an agent is no greater than its predecessor, the trajectory sequence will converge to a local optimum. However, if agents are always moving on locally optimal trajectories from themselves to their predecessors, we can obtain an non-increasing trajectory sequence but the end point is determined by the first agent and is not the best choice. We should have some freedom in choosing the final state instead of fixing it by simply catching up the leader's position. On the other hand, if at every updating step the follower is dealing with an optimal control problem with free final state, then it does not need the leader. The follower can determine the locally optimal trajectory only by its current state and the Δ, thus agents are totally independent. The aimlessly pursuing process will not let the follower "learn" from the leader and we can not guarantee the follower does better than the leader.

Based on the above consideration, we will let the follower "catch" the leader before the leader reaches the final state, i.e. the follower will solve an optimal control problem with fixed end point during $[t_k, t_k + i\delta)$, where $i\delta \leq T - \Delta$. After the leader reaches the final state, the follower will solve an optimal control problem with free final state. By dividing the time into two different phases - "catching up" and "free running" - the follower has the potential of "learning" from the leader as well as choosing the best final state. The trajectory sequence is expected to be gradually optimized through learning while it also benefits from the property of free final state.

As before, we define the cost of a segment of an optimal trajectory over $[t_0, t_0 + T]$ as:

$$\eta_{fs}(a, T, t_0) = Q(x(t_0 + T)) + \int_{t_0}^{t_0+T} g(x^*(t), \dot{x}^*(t), t)dt \tag{5.2}$$

where $x^*(t)$ minimize Eq. (5.1) with the restriction of $x^*(\tau) = a$. We here set up an algorithm similar to SLP, except replacing the step 2 to:

2. *When* $t = t_k + i\delta, i = 0, 1, 2, 3, \ldots,$ *calculate* $u_t^*(\tau)$ *such that* $f(x_k(\tau), u_t^*(\tau)) = \dot{x}_k^*(\tau)$, *where*
$$x_k^*(\tau) \text{ achieves } \begin{cases} \eta(x_k(t), x_{k-1}(t), \Delta, t, \Delta), & \tau \in [t, t+\Delta] \quad \textit{if } \Delta + i\delta < T \\ \eta_{fs}(x_k(t), t_k + T - t, t), & \tau \in [t, t_k + T] \quad \textit{otherwise} \end{cases}$$

If the final state is not free but restricted to a set, it should satisfy the final condition of

$$q(x(t_0 + T)) = 0 \qquad T \text{ is free} \tag{5.3}$$

For example, if the final state is located on a unit circle, the condition will be $\|x(t_0+T)\| = 1$. From optimal control, we have known that the best final time and state could be determined by the transversality condition.

We also define the cost of a segment of an optimal trajectory over $[t_0, t_0+T]$ as:

$$\eta_{fsg}(a, T, t_0) = Q(x(t_0+T)) + \int_{t_0}^{t_0+T} g(x^*(t), \dot{x^*}(t), t)dt \tag{5.4}$$

where $x^*(t)$ is the optimal trajectory for the cost in Eq. (5.1) while it satisfies that $x^*(t_0) = a, q(x^*(t_0+T) = 0$.

Of course we need both "catching up" and "free running" phases if applying local pursuit into such problems. We set up the algorithm as same as the CLP, except replacing the step 2 to

2. *Calculate $u_t^*(\tau)$ fro all $t \in [t_k, t_k+T]$ such that $f(x_k(\tau), u_t^*(\tau)) = \dot{x}_k^*(\tau)$, where* $x_k^*(\tau)$ *achieves* $\begin{cases} \eta(x_k(t), x_{k-1}(t), \Delta, t, \Delta), & \tau \in [t, t+\Delta] \quad \textit{if } t < t_k + T - \Delta \\ \eta_{fsg}(x_k(t), t_k + T - t, t), & \tau \in [t, t_k + T] \quad \textit{otherwise} \end{cases}$

The remaining work is to prove the optimality of the limiting trajectory obtained from the above algorithms. We may follow the similar steps as we do with SLP and CLP:

1. Proving the convergence of the cost incurred by the trajectories.

2. Proving the uniqueness of the limiting trajectory.

3. Proving the optimality of the composition of two segments of locally optimal trajectories.

4. Proving the local optimality of the limiting trajectory.

5.2 Convergence to Global vs Local Optimum

The limiting trajectory in local pursuit is determined by the parameters Δ, δ and the initial trajectory $x_0(t)$. As we shall see, different parameters may result in reaching different local optima. $x_0(t)$ is the initial trajectory generated by estimation or random exploration, and is not determined by the algorithms themselves.

There is an obvious trade-off in choosing Δ: large values of Δ may require significant demands on each agent's capabilities of sensing, communicating and computing, however, large Δ will also generally result in faster convergence and the ability of local pursuit to "escape" local optima. For the sake of space limitations we restrict our discussion to the following example[1].

[1]In this section, all the examples are deal with problems of minimizing the path length.

- If pursuit takes place on a surface with "holes" or "obstacles" and the initial feasible path winds around the obstacles. The iterated trajectories may converge to a global optimum instead of a local one with large Δ. For example, if the Δ is greater than 1/2 the perimeter of the largest circle that surrounds the holes and all agents run at unit speed, then the iterated trajectories converge to the global optimum.

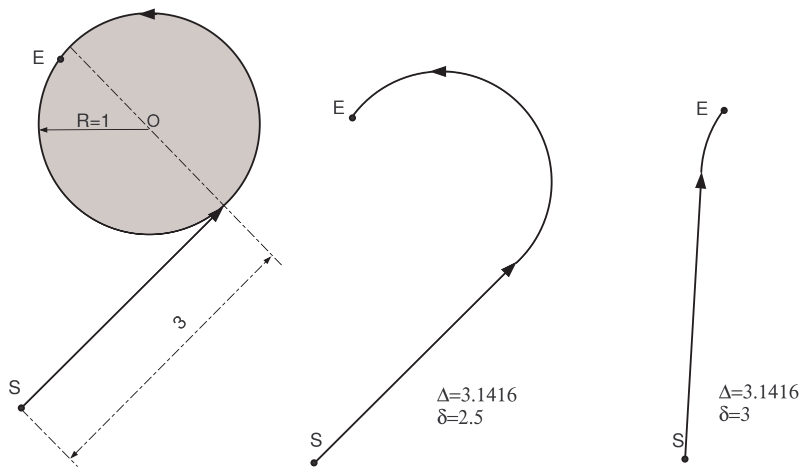


Figure 5.1: Larger δ may lead to a better result.

The δ is much easier to be adjusted because the only requirement for δ is that $0 \leq \delta < \Delta$. Nonetheless, there seems no simple relationship between the group's performance and δ. Smaller δ seemingly refers to more frequent updates and will bring better result, however, in fact it may lead to the local optimum instead of the global optimum. This can be seen from the following example.

- If pursuit takes place on a plane with a hole of unit radius, and each agent moves on unit speed. Let the first agent move counterclockwise along one local minimum from S to E, as illustrated in the left of Fig. 5.1. The Δ is set to be 3.1416 (a little more than π). Then the simulation shows that for some δ, e.g. $\delta = 2.5$, all the followers travel along the same trajectory as the leader's, and for some δ, e.g. $\delta = 3$, the limiting trajectory will converge to the global optimum. Here we see larger δ leads to better result. The two limiting trajectories in contrast are illustrated in Fig. 5.1.

We see that carefully selected parameters may lead to the global optimum. Directly determining the desired parameters that lead to desired local optimum seems difficult. However, given the parameters of an algorithm and a desired local optimum, we can investigate whether the trajectory sequence will converge to it. We will proceed using Lyapunov's method.

Recall that an agent's trajectory, $x_{k+1}(t)$, can be determined by the algorithm's parameters, if given its leader's trajectory $x_k(t)$. At every updating time, the locally optimal trajectory is determined by the starting state, the end state and Δ, so we can assume that the optimal trajectory minimizing the cost J is given by the mapping:

$$x^*(\tau) = h(a, b, \Delta, \tau) \qquad \tau \in [t, t + \Delta] \tag{5.5}$$

with the boundary condition $x(t) = a, x(t + \Delta) = b$. We assume that the mapping $h : \mathbb{D} \times \mathbb{D} \times \mathbb{R}^+ \times \mathbb{R} \to \mathbb{D} \times I$ (I is an time interval) defines a continuous trajectory. Given fixed Δ and δ (assume we start with SLP), and denoting the k^{th} trajectory as $x_k(t)$ ($t \in [t_k, t_k + T]$), then the $(k+1)^{th}$ trajectory is

$$x_{k+1}(t) = \begin{cases} h(x_{k+1}(t_{k+1} + j\delta), x_k(t_{k+1} + \Delta + j\delta), \Delta, t) & \text{if } t \in [t_{k+1}, t_{k+1} + i\delta] \\ h(x_{k+1}(t_{k+1} + i\delta), x_k(t_k + T), T - (i+1)\delta, t) & \text{otherwise} \end{cases} \tag{5.6}$$

where the integer i satisfies $i\delta \leq T - \Delta$ and $(i+1)\delta > T - \Delta$.

For simplicity, we can write Eq. (5.6) as

$$x_{k+1}(t) = f_p(x_k(t), \Delta, \delta, t) \qquad t \in [0, T] \tag{5.7}$$

where $f_p : \mathbb{D} \times [0, T] \times \mathbb{R}^+ \times \mathbb{R} \to \mathbb{D} \times [0, T]$ is a continuous function. This is similar to the state equation of a dynamic system, if we think every trajectory as a state in the space of trajectories ($\mathbb{D} \times [0, T]$).

Noticing that Lyapunov's method is a commonly applied technique in analyzing the convergence properties of dynamic systems, we plan to set up a Lyapunov function in the space of trajectories. The constructed Lyapunov should satisfy the following condition:

$$V(x^*(t)) = q \text{ and } V(x_k(t)) > q \text{ in } \mathbb{D} \times [0, T] - \{x^*(t)\} \tag{5.8}$$

$$V(x_{k+1}(t)) - V(x_k(t)) \leq -\rho_k < 0 \text{ for } x_k(t) \in \mathbb{D} \times [0, T] - \{x^*(t)\} \tag{5.9}$$

where $x^*(t)$ is the predetermined local optimum and $\rho_k \to 0$ only if $x_k(t) \to x^*(t)$. If we can find such a Lyapunov function, we can conclude that the trajectory sequence generated by local pursuit and started with an initial trajectory $x_0(t) \in \mathbb{D} \times [0, T]$ will converge to $x^*(t)$. Furthermore, by finding a region in the space of trajectories where the Lyapunov function satisfies the above condition of Eq. (5.8),(5.9) and is bounded above, we are expected to find the region of attraction of this local optimum.

5.3 Pursuit with Noisy Measurements

In the real world, sensors and actuators embedded in robots are not perfect and the operation of them is often distorted by noise. There are a number of key points to understand the uncertainty [21]:

- Sensors only deliver uncertain values in practice. At best they deliver an approximation to what they are measuring. The disturbance in environments, difference between physical parts and measuring mechanism are also bringing unexpected errors for every sensor. Moreover, sensors do not deliver direct descriptions of the world. They do not identify the objects and separate the effects due to their own motion and objects' motion. Therefore we can hardly obtain an accurate model for a real sensor.

- Commands to actuators can have uncertain effects. Many layers of refinement may be performed before high level action commands become appropriate motor currents, each may bring uncertainty. Depending on the hardware and software accuracy, errors could accumulate rapidly. These uncertainties make it difficult to model actuators accurately.

What we want to investigate is the collective behavior of the system when the measurements made by agents are subjected to noise. We would like to develop algorithms that work not only well but also robustly. For the sake of simplicity, we may consider the noise of sensors and actuators together and model the noise in a generic, abstract context as

$$\hat{x}(t) = x^*(t) + \xi(x)\omega(t) \tag{5.10}$$

where $x^*(t)$ is the actually optimal trajectory, $\xi(x)$ is a real valued function and $\omega(t)$ is a white Gaussian process with mean $\bar{\omega}$ and variance Σ^2. What we are interested in is to investigate the limiting trajectory and determine its distribution.

Another source of uncertainty comes from the estimation of optima, when precise solutions to locally optimal trajectories is impossible to obtain even though all measurement and models are perfect. For example, for an uneven terrain that can not be described by any existing geometric objects, it is hardly to obtain an analytical solution to the geodesics on it. However, sometimes we can estimate the solution with bounded error through numerical methods or other simple rules, by investigating properties of the environment and the optimal solution. The error of local estimate is related to the "following distance" between the leader and the follower: the smaller the distance is, more precise the estimate. So in this case the locally optimal trajectories that agents obtain are as follow.

$$\hat{x}(t) = x^*(t) + \varepsilon(t) \qquad \|\varepsilon(t)\|_\infty < \mathcal{L} \tag{5.11}$$

and we are interested to find the range of the limiting trajectory's error. If the error of the limiting trajectory is bounded (depends on $\mathcal{L}$), then we have found a method to transform the local trajectories' error to the entire trajectory's error.

In order to proceed with this, the following steps will be considered:

1. Making a model of locally optimal trajectories at updating times, as Eq. (5.10),(5.11) did.
2. Investigating the evolution of each pair of leader and follower under noisy measurements and the evolution of the trajectory's error through the pursuit process.
3. Finding the error of the limiting trajectory.

5.4 Application in Numerical Computation of Optimal Control

The algorithms stated here can potentially lead to advances in numerical computing of optimal trajectories for control systems. Numerical methods, including the Newton's method and gradient methods, are commonly applied optimization methods. An obvious drawback of ordinary numerical methods is that they need large amount of calculation for they are optimizing the result iteratively.

For example, the multiple shooting method is widely used in difficult applications, e.g. fuel optimization problem for spaceships [9]. Proceeding formally to multiple shooting method, as Betts summarized in [8], "the fundamental idea of multiple shooting is to break the trajectory into shorter pieces or segments". The time domain is broken into smaller intervals of the form $t_0 < t_1 < \cdots < t_M = t_f$. The initial value for the dynamic variable at the beginning of each segment is denoted as ν_j for $j = 0, 1, \ldots, (M-1)$ and the variable obtained through solving system equation from t_j to t_{j+1} is denoted as $\bar{\nu_j}$. The nonlinear programming (NLP) variables are defined as $x = [\nu_0, \nu_1, \ldots, \nu_{M-1}]$. And the constraints for NLP are

$$c(x) = \begin{bmatrix} \nu_1 - \bar{\nu_0} \\ \nu_2 - \bar{\nu_1} \\ \vdots \\ \phi(\nu_M, t_f) \end{bmatrix} = 0$$

where $\phi(\nu_M, t_f) = 0$ is the boundary condition. The number of NLP variables and constrains is $n = n_\nu M$ where n_ν is the dimension of dynamic variable ν and M is the number of segments [8] [13]. Thereafter the problem to minimize cost function $F(x)$ can been solved by introducing the Lagrangian

$$L(x, \lambda) = F(x) - \lambda^T c(x)$$

Necessary conditions for the variable $[x^*, \lambda^*]$ to be an optimum are defined by

$$\begin{aligned} \nabla_x L(x, \lambda) &= 0 \\ \nabla_\lambda L(x, \lambda) &= 0 \end{aligned}$$

When proceeding with the iterating process, the dimension of Jacobian matrix ∇_x is $n \times n = n_\nu M \times n_\nu M$. Here we have seen that the number of segments involved in the calculation affects the "degree of labor-consumption" at least in the order of $O(n^2)$. Moreover, increasing the variable size will lead to more iterating steps. If fewer segments were introduced during calculation processes, the complexity of computing can be decreased.

Another obvious example of "more time segments lead to increased complexity" is the dynamic programming. Bellman introduced the Hamilton-Jacobi-Bellman (HJB) equation to describe the optimal control $u^*(x, t)$ as well as the cost-to-go function $J^*(x, t)$ for all possible initial conditions [17, 14]. The HJB theory plays an important role in the field of optimal control because it provides sufficient condition for optimality as opposed to the necessary condition obtained from ordinary optimization methods [8]. However, the drawback of dynamic programming is the "curse of dimensionality", as Bellman himself calls it. Even dealing with a moderately complicated problem will involve an enormous amount of storage [15]. This drawback of dynamic programming could be seen from the discrete example of shortest path problem [16] [17], as illustrated by the trellis diagram in Fig. 5.2. The worst case will involve investigating $n_\nu^2 M$ paths and storing $n_\nu M$ data if proceeding backward from the end point to the starting point, where n_ν is the dimension of state x and M is the number of segments. Operation using dynamic programming with large M is often unfeasible due to the agent's limited physical memory.

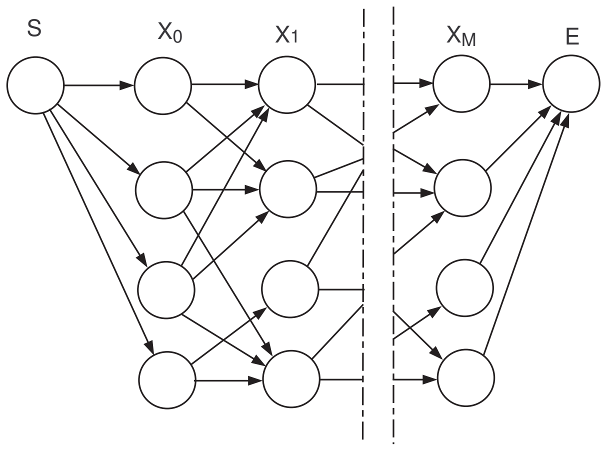


Figure 5.2: The trellis diagram of shortest path problem.

We would like to utilize numerical methods with less computing complexity. One idea is to decrease the number of time segments in operational processes. Fewer segments mean that optimization processes can only be executed in smaller regions, which coincides with the idea of finding optima within small regions, as stated before. Therefore we plan to apply local pursuit in numerical methods and investigate the potential advantages that appear, such as the decrease of physical requirements for each agent and the "degree of labor-consumption" for the group. To complete the argument, we should consider the following steps:

1. Applying local pursuit in numerical methods to solve some optimal control problems. Investigate a single updating process, determine the requirement for an individual agent to proceed the algorithm, e.g. the size of storage, the complexity of computing.

2. Trying to find the appropriate iterative times to reach the satisfying result, e.g. to determine the k so that

$$\|x_k(t) - x^*(t)\|_\infty \le \varepsilon \tag{5.12}$$

3. Investigating the requirements and computing complexity in numerical method ordinarily applied in the same problems.

4. Comparing the two kinds of numerical methods.

5.5 Other Algorithms Inspired by Biology

Besides ants, other social insects, e.g. worker honey bees, have shown us a lot of group activities with amazing coordinated behaviors. The intrinsic mechanism has been partly revealed by some effective models of such activities. We are considering ways to "borrow" the rules that govern behaviors of insects and to develop additional biologically inspired algorithms for problems in engineering. Some potential topics are as follows:

- The foraging activities of worker honey bees [1] can provide us with some clues on solving the resource allocating problems, which has numerous applications in engineering, economics and research operation, e.g. routing a group of taxis to pick up and deliver passengers whose appearances are dynamic or random, arranging a limited number of robots to execute several manufacturing processes.

- The work of [3] presented a model of how ants select ongoing foraging zones. According to this model, each ant has the uniform distribution over all foraging zones at first. Assume the probability of foraging zone i at time t is $P_i(t)$. If at time t, the ant finds food in zone i, then the probability $P_i(t+1) = P_i(t) + \min(P^+, 1 - P_i(t))$, where

P^+ is a constant indicating the relative importance of "learning". If not, then the probability $P_i(t+1)$ will be decreased. By this mechanism, both an individual ant and a colony of ants will evolve into optimal spatial distribution over foraging zones - getting maximum food when the appearance of food at each zone is random and unknown to the ants. In engineering, this method of "learning" is helpful, especially when there exist unknown factors. For example, we may want to use limited number of controllers to stabilize multiple plants. However, each plant has the unknown distribution of deviating from its equilibrium position and we want to minimize the sum of deviations. It is promising that we can let the controllers learn the distribution of each plant and develop decentralized rules for each controller.

In order to successfully complete the proposed research, the following are specific steps to be taken:

1. Finding some engineering or economic problems with similar properties to a social insect activity and constructing an effective model of insect activities. Many works have discussed models of social behavior in insects. We will stress those that appear to have the simplest rules.

2. Abstracting the rules that govern communication and motion behaviors of insects and embedding it into the artificial collectives in order to solve the proposed tasks.

3. Showing the effectiveness of proposed algorithm by analysis or simulation.

Appendix A

Proofs

A.1 Preliminaries

The following facts can be derived easily from the properties of optimal trajectories and are helpful in future argument.

Facts : *Let η, C, η_F as defined in Eq. (3.4),(3.5),(3.8), $x_k(t)$ be a trajectory of Eq. (3.1) and $x^*(t)$ an optimal trajectory of Eq. (3.3) or Eq. (3.7). Then, the following properties hold:*

1. $\eta(a, b, T, t_0, \sigma) \leq C(x_k, t_0, \sigma)$ *with any* $x_k(t_0) = x^*(t_0), x_k(t_0 + \sigma) = x^*(t_0 + \sigma)$ *where* $x^*(t)$ *satisfies Eq. (3.3).*
2. $\eta(a, c, T, t_0, T) \leq \eta(a, b, \sigma, t_0, \sigma) + \eta(b, c, T - \sigma, t_0 + \sigma, T - \sigma)$
3. $C(x_k, t_0, T) = C(x_k, t_0, \sigma) + C(x_k, t_0 + \sigma, T - \sigma)$
4. $\eta_F(a, b, t_0, \sigma) \leq \eta(a, b, T, t_0, \sigma)$
5. $\eta(a, b, T, t_0, \sigma) = C(x^*, t_0, \sigma)$ *where* $x^*(t)$ *satisfies Eq. (3.3).*

A.2 Proof of Lemma 4.1

It is enough to show the cost of the iterated trajectories is non-increasing with k. Consider the pursuing process between the $(k-1)^{th}$ and k^{th} agents. As shown in Fig. A.1, the dotted line, denoted by $x_{k-1}(t)$ on $[t_{k-1}, t_{k-1} + T]$, indicates the leader's path. The solid lines, denoted by $x_k(t)$, are the trajectories of the "follower", and the dashed lines, noted by $\hat{x}_k(t)$, are the planned trajectories, as described before. And we use $\tilde{x}(t)$ to denote the trajectory that the follower copies from the leader's trajectory but with a delay of time Δ, i.e. $\tilde{x}_k(t+\Delta) = x_{k-1}(t)$. Therefore the cost along it must be same to the cost along the leader's.

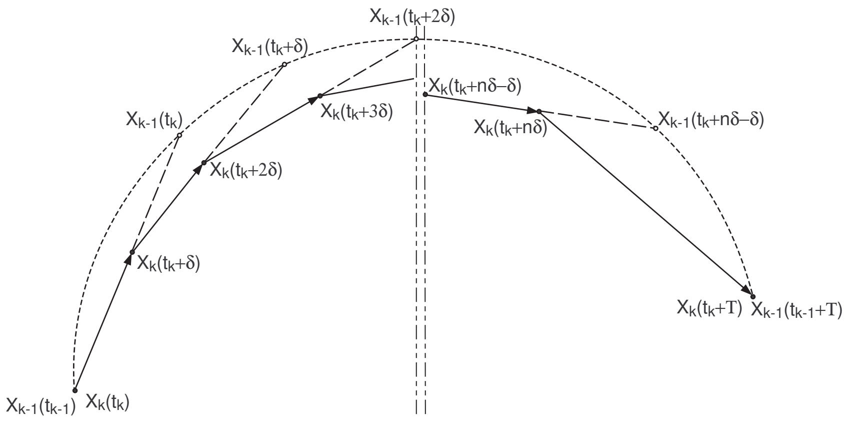


Figure A.1: Sketch of Sampled Local Pursuit

The follower leaves the starting state at time t_k, while the leader leaves it at time t_{k-1}, where $t_k = t_{k-1} + \Delta$. For $t \in [t_k, t_k+\delta]$, the follower moves on an optimal trajectory from state $x_k(t_k)$ to $x_{k-1}(t_k)$ over Δ units of time. Thus from Fact 1:

$$\begin{aligned} \eta(x_k(t_k), x_{k-1}(t_k), \Delta, t_k, \Delta) &\leq C(\tilde{x}_k, t_k, \Delta) \\ &= C(x_{k-1}, t_{k-1}, \Delta) \end{aligned} \tag{A.1}$$

The right-hand side is the cost along the leader's path for the first Δ units of time, the left-hand side is the optimal cost from $x_k(t_k)$ to $x_{k-1}(t_k)$.

At time $t_k+\delta$ the follower reaches the state $x_k(t_k+\delta)$. Recalling that the trajectory drvien by $u^*_{t_k}(\tau)$ is optimal from $x_k(t_k)$ to $x_{k-1}(t_k)$ and from Fact 3, we can divide the cost into two parts, one is actual and the other is planned[1], i.e.

$$\begin{aligned} &\eta(x_k(t_k), x_{k-1}(t_k), \Delta, t_k, \Delta) \\ = \; &\eta(x_k(t_k), x_{k-1}(t_k), \Delta, t_k, \delta) + \eta(x_k(t_k+\delta), x_{k-1}(t_k), \Delta-\delta, t_k+\delta, \Delta-\delta) \end{aligned} \tag{A.2}$$

From (A.1),(A.2):

$$\begin{aligned} &\eta(x_k(t_k), x_{k-1}(t_k), \Delta, t_k, \delta) \\ \leq \; &C(x_{k-1}, t_{k-1}, \Delta) - \eta(x_k(t_k+\delta), x_{k-1}(t_k), \Delta-\delta, t_k+\delta, \Delta-\delta) \end{aligned} \tag{A.3}$$

At time $t_k+\delta$, the follower updates its trajectory to catch the leader at its new location $x_k(t_k+\delta)$. For this trajectory is optimal from $x_k(t_k+\delta)$ to $x_{k-1}(t_k+\delta)$ over time Δ, any

[1]These two pieces are both optimal with respect to their corresponding end points.

path $x_k(t)$ ($t \in [t_k + \delta, t_k + \delta + \Delta]$) that is from $x_k(t_k + \delta)$ to $x_{k-1}(t_k + \delta)$ over time Δ and passes through $x_{k-1}(t_k)$ at time $t_k + \Delta = t_k + \delta + \Delta - \delta$ has equal or more cost. From Fact 2 follows:

$$\begin{aligned}
& \eta(x_k(t_k+\delta), x_{k-1}(t_k+\delta), \Delta, t_k+\delta, \Delta) \\
\leq\ & \eta(x_k(t_k+\delta), x_{k-1}(t_k), \Delta-\delta, t_k+\delta, \Delta-\delta) + \eta(x_{k-1}(t_k), x_{k-1}(t_k+\delta), \delta, t_k+\Delta, \delta) \\
\leq\ & \eta(x_k(t_k+\delta), x_{k-1}(t_k), \Delta-\delta, t_k+\delta, \Delta-\delta) + C(\tilde{x}_k, t_k+\Delta, \delta) \\
=\ & \eta(x_k(t_k+\delta), x_{k-1}(t_k), \Delta-\delta, t_k+\delta, \Delta-\delta) + C(x_{k-1}, t_k, \delta) \qquad \text{(A.4)}
\end{aligned}$$

We can also divide this cost into a realized part and a planned one, i.e.

$$\begin{aligned}
& \eta(x_k(t_k+\delta), x_{k-1}(t_k+\delta), \Delta, t_k+\delta, \Delta) \\
=\ & \eta(x_k(t_k+\delta), x_{k-1}(t_k+\delta), \Delta, t_k+\delta, \delta) + \eta(x_k(t_k+2\delta), x_{k-1}(t_k+\delta), \Delta-\delta, t_k+2\delta, \Delta-\delta) \qquad \text{(A.5)}
\end{aligned}$$

From $(A.1) \sim (A.5)$, we obtain

$$\begin{aligned}
& C(x_k, t_k, 2\delta) \\
=\ & \eta(x_k(t_k), x_{k-1}(t_k), \Delta, t_k, \delta) + \eta(x_k(t_k+\delta), x_{k-1}(t_k+\delta), \Delta, t_k+\delta, \delta) \\
\leq\ & C(x_{k-1}, t_{k-1}, \Delta) + C(x_{k-1}, t_k, \delta) - \eta(x_k(t_k+2\delta), x_{k-1}(t_k+\delta), \Delta-\delta, t_k+2\delta, \Delta-\delta) \\
=\ & C(x_{k-1}, t_{k-1}, \Delta+\delta) - C(\hat{x}_k, t_k+2\delta, \Delta-\delta) \qquad \text{(A.6)}
\end{aligned}$$

where $\eta(x_k(t_k+2\delta), x_{k-1}(t_k+\delta), \Delta-\delta, t_k+2\delta, \Delta-\delta) = C(\hat{x}_k, t_k+2\delta, \Delta-\delta)$ is from the fact that the planned trajectory is optimal.

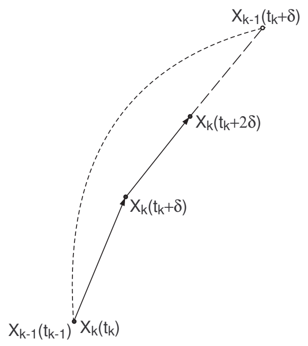


Figure A.2: First two steps in sampled local pursuit

We repeat this procedure until $t = t_k + n\delta$ where $\Delta + (n-1)\delta < T$ and $\Delta + n\delta \geq T$. This choice of n means that the leader has not reached the final state, and

$$\begin{aligned}
C(x_k, t_k, n\delta) &= \sum_{i=0}^{n-1} \eta(x_k(t_k+i\delta), x_{k-1}(t_k+i\delta), \Delta, t_k+i\delta, \delta) \\
&\leq C(x_{k-1}, t_{k-1}, \Delta+(n-1)\delta) - C(\hat{x}_k, t_k+n\delta, \Delta-\delta) \qquad \text{(A.7)}
\end{aligned}$$

When $t \in [t_k + n\delta, t_k + T]$, the leader reaches the final state and stays static. During this time period, no matter how many times the follower updates its movement, it will move on the same path that was determined at time $t = t_k + n\delta$. This path, which is indicated by the last solid line in Fig. A.1, is locally optimal between the states $x_k(t_k + n\delta)$ and $x_k(t_k + T)$ over $T - n\delta$ units of time. Therefore

$$\begin{aligned} & C(x_k, t_k + n\delta, T - n\delta) \\ = \; & \eta(x_k(t_k + n\delta), x_{k-1}(t_{k-1} + T), T - n\delta, t_k + n\delta, T - n\delta) \\ \leq \; & C(\hat{x}_k, t_k + n\delta, \Delta - \delta) + C(x_{k-1}, t_k + (n-1)\delta, T - (n-1)\delta - \Delta) \end{aligned} \tag{A.8}$$

From $(A.7) \sim (A.8)$, we obtain

$$\begin{aligned} C(x_k, t_k, T) & \leq C(x_{k-1}, t_{k-1}, \Delta + (n-1)\delta) + C(x_{k-1}, t_k + (n-1)\delta, T - (n-1)\delta - \Delta) \\ & = C(x_{k-1}, t_{k-1}, T) \end{aligned} \tag{A.9}$$

We have shown that cost incurred by the follower is no greater than the leader's. Writing $C_k = C(x_k, t_k, T)$ in convenience, we can see that $C_k \leq C_{k-1}$. Obviously C_k is bounded below if there exits an optimal trajectory from the starting state to the target state. Hence we conclude that

$$\lim_{k \to \infty} C_k = C \tag{A.10}$$

A.3 Proof of Lemma 4.2

Suppose there exist more than one limiting trajectory, and suppose $x_1(t)$ and $x_2(t)$ are two possibilities. $x_1(t)$ differs from $x_2(t)$ for $t \in [t_1, t_2] \cup [t_3, t_4] \ldots$. From Lemma 4.1 these two trajectories must have the same cost.

Let the leader $x_{k-1}(t)$ travel along $x_1(t)$, while the follower $x_k(t)$ travels along $x_2(t)$. If no update occurs during $[t_1, t_2]$, $x_2(t)$ has less cost during $[t_1, t_2]$ because the follower moves along $x_2(t)$ and the local optimum is unique. Same arguments on other different time periods lead to the face that the whole cost along $x_2(t)$ is less than $x_1(t)$ if no update occurs during $t \in [t_1, t_2] \cup [t_3, t_4] \ldots$, which contradicts to the fact that two trajectories have the same cost.

Next, assume only one update occurs during $[t_1, t_2]$, as Fig. A.3 indicates. Separate the curves during $[t_1, t_2]$ into several segments (the meaning of different curve style is the same as in Lemma 4.1), and indicate the cost along curve i as C_i. From the uniqueness of local optimum, we have $C_1 + C_5 < C_3$ and $C_2 < C_5 + C_4$. Hence $C_1 + C_2 < C_3 + C_4$, which means $x_2(t)$ has less cost than $x_1(t)$ during $[t_1, t_2]$.

If there are multiple updates during $[t_1, t_2]$, we can see that the updates does not change the fact that cost along $x_2(t)$ is less than $x_1(t)$. Hence we still get the result that the cost along $x_2(t)$ is less than $x_1(t)$ for $t \in [t_1, t_2]$, no matter how many updates occur.

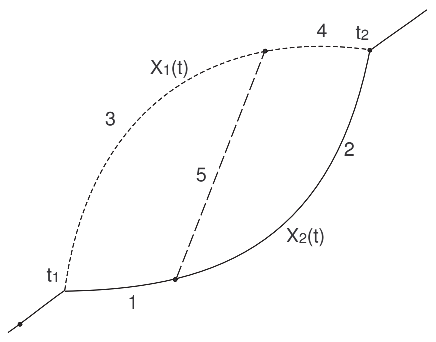


Figure A.3: There is one update between two trajectories

Iterating on more different time periods leads to the fact that the whole cost along $x_2(t)$ must be less than $x_1(t)$. We also obtain contradiction.

A.4 Proof of Lemma 4.3

Let the leader move along the limiting trajectory $x_\infty(t)$, suppose it is the $(k-1)^{th}$ agent. From Lemma 4.2, the limiting trajectory means that $x_{k-1}(t) = x_k(t+\Delta)$ for $\forall t \in [t_k, t_k+T]$.

At first we claim that in the time interval $[t_k+\delta, t_k+\Delta]$, the planned trajectory agrees with the realized one, i.e. $\hat{x}_k(t) = x_k(t), t \in [t_k+\delta, t_k+\Delta]$. Suppose that $\hat{x}_k(t) \neq x_k(t)$ for some $t \in [t_k+\delta, t_k+\Delta]$. Because $\hat{x}(t)$ is optimal from $x_k(t_k+\delta)$ to $x_k(t_k+\delta+\Delta)$, the trajectory

$$\bar{x}(t) = \begin{cases} \hat{x}_k(t) & t \in [t_k+\delta, t_k+\Delta) \\ x_k(t) & t \in [t_k+\Delta, t_k+\delta+\Delta] \end{cases}$$

has less cost than the trajectory $x_k(t)$ ($t \in [t_k+\delta, t_k+\delta+\Delta]$) , which is updated by the follower at the time $t = t_k+\delta$ and is supposed to be optimal from $x_k(t_k+\delta)$ to $x_k(t_k+\delta+\Delta)$. Thus there is a contradiction. Hence we obtain $\hat{x}_k(t) = x_k(t)$ for $\forall t \in [t_k+\delta, t_k+\Delta]$. Same arguments could be applied in other time periods.

$\bar{x}(t)$ is smooth for $t \in [t_k, t_k+\Delta]$ because the locally optimal trajectory is smooth, and $x_k(t)$ is smooth for $t \in [t_k+\delta, t_k+\delta+\Delta]$(second update step) because of the same reason. And we know $\hat{x}_k(t) = x_k(t)$ for $\forall t \in [t_k+\delta, t_k+\Delta]$. Thus the actual trajectory $x_k(t)(t \in [t_k, t_k+2\delta])$ is smooth. Continuing on this argument leads to the result that the whole trajectory $x_k(t)$ ($t \in [t_k, t_k+T]$) is smooth.

A.5 Proof of Lemma 4.4

We rewrite the lemma to that if $x^*(t)$ $(t \in [0, t_1 + \Delta_1])$ and $x^*(t)$ $(t \in [t_1, T])$ are two locally optimal trajectories and Condition 4.1 is satisfied, where $0 < t_1 < t_1 + \Delta_1 < T$, then the trajectory $x^*(t), t \in [0, T]$ is a local minimum.

We take $0 < \Delta \leq \Delta_1$. From principle of optimality, we obtain that $x^*(t)(t \in [0, t_1 + \Delta])$ and $x^*(t)(t \in [t_1, T])$ are two locally optimal trajectories with respect to their corresponding end points.

Suppose that $x^*(t)(t \in [0, T])$ is not the local minimum, there must exist an $\epsilon < \varepsilon$ and another optimum $x(t) \in \mathbb{D} \times [0, T]$ satisfying that $\|x(t) - x^*(t)\|_\infty < \epsilon$ and $C(x(t), 0, T) < C(x^*(t), 0, T)$, as Fig. A.4 shows.

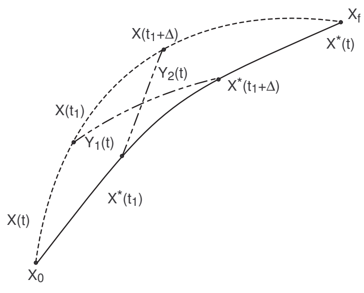


Figure A.4: Overlapped local minimums lead to the local minimum overall

Construct two optimal trajectories $y_1(t), y_2(t), t \in [t_1, t_1 + \Delta]$ connecting $x(t)$ and $x^*(t)$ such that $x^*(t_1) = y_2(t_1), x^*(t_1 + \Delta) = y_1(t_1 + \Delta), x(t_1) = y_1(t_1), x(t_1 + \Delta) = y_2(t_1 + \Delta)$. From principle of optimality, $x^*(t)$ and $x(t)$ $(t \in [t_1, t_1 + \Delta])$ are both optimal trajectories with respect to their corresponding end points. Now with the condition of Eq. (4.1), we obtain

$$\begin{aligned} C(y_1(t), t_1, \Delta) &< C(x(t), t_1, \Delta) + \mathcal{L}\Delta \\ C(y_2(t), t_1, \Delta) &< C(x^*(t), t_1, \Delta) + \mathcal{L}\Delta \end{aligned} \tag{A.11}$$

For $x^*(t)$ $(t \in [0, t_1 + \Delta])$ and $x^*(t)$ $(t \in [t_1, T])$ are two unique local optimal trajectories, we have

$$\begin{aligned} C(x^*(t), 0, t_1) + C(x^*(t), t_1, \Delta) &< C(x(t), 0, t_1) + C(y_1(t), t_1, \Delta) \\ C(x^*(t), t_1, \Delta) + C(x^*(t), t_1 + \Delta, T - t_1 - \Delta) &< C(x(t), t_1 + \Delta, T - t_1 - \Delta) + C(y_2(t), t_1, \Delta) \end{aligned} \tag{A.12}$$

Combining (A.11) and (A.12) leads to

$$C(x^*(t), 0, T) + C(x^*(t), t_1, \Delta) < C(x(t), 0, T) + C(x^*(t), t_1, \Delta) + 2\mathcal{L}\Delta$$

which could be derived as

$$C(x^*(t), 0, T) < C(x(t), 0, T) + 2\mathcal{L}\Delta \tag{A.13}$$

$C(x(t), 0, T)$ is assumed to be less than $C(x^*(t), 0, T)$, but if we take

$$0 < \Delta < \frac{C(x^*(t), 0, T) - C(x(t), 0, T)}{2\mathcal{L}}$$

Therefore Eq. (A.13) can not be true. There is a contradiction because Δ could be set to be arbitrarily small. Hence follows the conclusion that $x^*(t)$ ($t \in [0, T]$) must be the local minimum.

A.6 Proof of Lemma 4.5

Suppose that $t_k + \lambda + \Delta < T$. As Fig. A.5 indicated, the follower moves on the locally optimal trajectory $x_k(t)(t \in [t_k + \lambda, t_k + \lambda + \Delta])$ at time $t_k + \lambda$. Define a function $G : \mathbb{D} \times [0, T] \times \mathbb{R}^+ \to \mathbb{D} \times [0, T]$ to represent the new trajectory, denoted as $G(\lambda, x_{k-1}(t))$. The cost along the follower's trajectory is

$$\begin{aligned} C(x_k, t_k, T) &= C(x_k, t_k, \lambda) + \eta(x_k(t_k + \lambda), x_{k-1}(t_k + \lambda), \Delta, t_k + \lambda, \Delta) + C(x_k, t_k + \Delta, T - \Delta) \\ &\leq C(x_{k-1}, t_{k-1}, \lambda) + C(x_{k-1}, t_{k-1} + \lambda, \Delta) + C(x_{k-1}, t_{k-1} + \lambda + \Delta, T - \lambda - \Delta) \\ &= C(x_{k-1}, t_{k-1}, T) \end{aligned} \tag{A.14}$$

Same argument could be applied for the case where $t_k + \lambda + \Delta \geq T$.

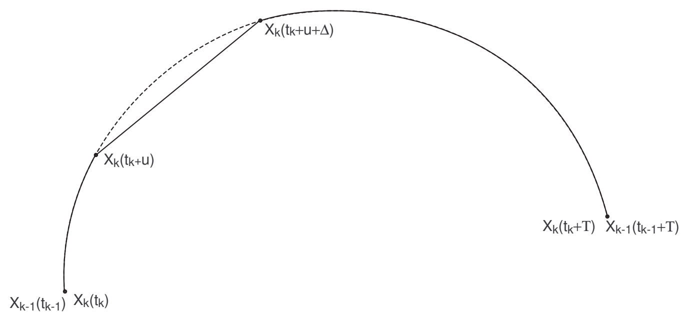


Figure A.5: The cost is decreased with a single update.

A.7 Proof of Lemma 4.6

Suppose the cost along the leader's trajectory $x_{k-1}(t)$ ($t \in [t_{k-1}, t_{k-1} + T]$) is C_{k-1}. Set up a

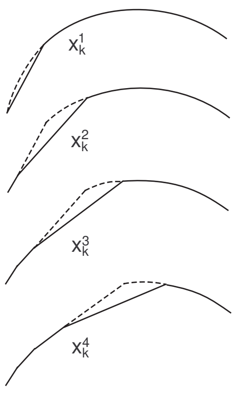


Figure A.6: Trajectory Sequence of $x_k^i(t)$.

trajectory sequence $x_k^i(t)$ $(t \in [t_k, t_k + T]), i = 1, 2 \dots$ with the corresponding cost of C_k^i. Let $x_k^0(t) = x_{k-1}(t)$ and $x_k^i(t) = G((i-1)\delta, x_k^{i-1}(t))$, as Fig. A.6 indicates, where G is defined in the proof of Lemma 4.5.

According to Lemma 4.5,

$$C_k^i \leq C_k^{i-1} \Rightarrow C_k^\infty \leq C_k^0 = C_{k-1}$$

with $\delta \geq 0$.

Let $\delta = T/i$, then $\delta \to 0$ as $i \to \infty$. And now the trajectory $x_k^i(t)$ is exactly under the same updating process as in the continuous local pursuit. Therefore we obtain the follower's cost $C_k = C_k^\infty \leq C_{k-1}$. Since the sequence $\{C_k\}$ is non-increasing, surely it will converge to a limit.

Bibliography

[1] S. Camazine, etc.(Eds), Self-Organization in Biological Systems, Princeton University Press, 2001.

[2] J. M. Pasteels and J-L. Deneubourg(Eds), From individual to collective behavior in social insects, Les Treilles Workshop, Basel, Boston, 1987.

[3] J-L. Deneubourg, S. Goss, J. M. Pasteels, D. Fresneau and J-P. Lachaud, Self-organization mechanisms in ant societies (II): learning in foraging and division of labor, in: J. M. Pasteels and J-L. Deneubourg(Eds), From individual to collective behavior in social insects, Les Treilles Workshop, Basel, Boston, 1987, pp. 177-196.

[4] D. M. Gordon, Ants at work, The Free Press, New York, 1999.

[5] H. K. Khalil, Nonlinear Systems, Prentice Hall, New jersey, 2002.

[6] E. K. P. Chong and S. H. Żak, An introduction to optimization, Wiley, New York, 1996.

[7] D. J. Wilde, Optimum seeking methods, Prentice-Hall, Englewood Cliffs, N.J. 1964.

[8] J. T. Betts, Survey of numerical methods for trajectory optimization, Journal of Guidance, Control and Dynamics, Vol. 21, No. 2, Mar. - Apr. 1998.

[9] R. S. Nah, S. R. Vadali and E. Braden, Fuel-optimal, low-thrust, three-dimensional earth-mars trajectories, Journal of Guidance, Control, and Dynamics, Vol. 24, No. 6, Nov.-Dec., 2001.

[10] D. Kincaid and W. Cheney, Numerical Analysis, Brooks/Cole Publishing Company, Pacific Grove, California, 1991.

[11] R. Kress, Numerical Analysis, Springer-Verlag New York Inc, New York, 1998.

[12] N. S. Bakhvalov, On the optimization of numerical algorithms, in: B. Bojanov, H. Woźniakowski(Eds), Optimal Recovery, Nova Science Publishers, Inc. 1992, pp. 1-58.

[13] H. B. Keller, Numerical methods for two-point boundary value Problems, Blaisdell Publishing Company, Waltham, Massachusetts, 1968.

[14] G. L. Nemhauser, Introduction to dynamic programming, John Wiley and Sons, Inc., New York, 1966.

[15] A. E. Bryson, Jr., Y. C. Ho, Applied optimal control: optimization, estimation and control, Hemisphere Pub. Corp., Washington, 1975.

[16] G. D. Forney, Jr., The viterbi algorithm, Proceedings of the IEEE, vol.61, no.3, March, 1973, pp. 268-278.

[17] D. P. Bertsekas, Dynamic programming and optimal control, Volume I, Athena Scientific, Belmont, Massachusetts, 2000.

[18] D. Hristu-Varsakelis, Robot formations: Learning minimum-length paths on uneven terrain, Proceedings of the 8th IEEE Mediterranean Conference on control and Automation, 2000.

[19] D. Hristu-Varsakelis, P. Krishnaprasad, S. Andersson, F. Zhang, P. Sodre, L. D'Anna, The MDLe engine: a software tool for hybrid motion control, Tech. Rep. TR2000-54, Institute for systems Research, Oct. 2000.

[20] L. Edelstein-Keshet, Trail following as adaptable mechanism for popular behavior, in: R. Murphey, P. M. Pardalos(Eds), Animal groups in three dimensions, Cambridge University Press, Cambridge, U.K., 1997, pp. 282-300.

[21] R. A. Brooks, Artificial life and real robots, Proceeding of the First European Conference on Artificial Life, MIT Press/Bradford Books, Cambridge, 1992, pp. 3-10.

[22] R. A. Brooks and A. M Flynn, Fast, cheap and out of control: a robot invasion of the solar system, Journal of the British Interplanetary Society, Vol, 42, 1989, pp. 478-485.

[23] R. Murphey, P. M. Pardalos(Eds), Cooperative control and optimization, Kluwer Academic Publishers, 2002.

[24] T. Vicsek, A. Czirok, E. B. Jacob, I. Cohen, and O. Schochet, Novel type of phase transitions in a system of self-driven particles, Physical Review Letters, Vol. 75, 1995, pp. 1226-1229.

[25] A. Jadbabaie, J. Lin and A. S. Morse, Coordination of groups of mobile autonomous agents using nearest neighbor rules, IEEE Transactions on Automatic Control, Vol. 48, No. 6, June 2003.

[26] A. M. Bruckstein, Why the ant trails look so straight and nice, The Mathematical Intelligencer, 15(2), 1993, pp. 59-62.

[27] A. M. Bruckstein, C. L. Mallows, and I. A. Wagner, Probabilistic pursuits on the grid, The American Mathematical Monthly, Vol. 104, No. 4, April, 1997, pp. 323-343.

[28] I. A. Wagner, M. Lindenbaum, A. M. Bruckstein, Distributed covering by ant-robots using evaporating traces, IEEE Transactions on Robotics and Automation, Vol. 15, No. 5, 1999, pp. 918-933.

[29] K. Passino, M. Polycarpou, D. Jacques, M. Pachter, Y. Liu, Y. Yang, M. Flint and M. Baum, Cooperative control for autonomous air vehicles, in: R. Murphey, P. M. Pardalos(Eds), Cooperative control and optimization, Kluwer Academic Publishers, 2002, pp. 233-272.

[30] R. Kurazume, S. Hirose, S. Nagata and N. Sashida, Study on cooperative positioning system(Basic principle and measurement experiment), Proceeding of the 1996 IEEE International Conference in Robotics and Automation, Vol. 2, Minneapolis, MN, April, 1996, pp. 1421-1426.

[31] R. Kurazume and S. Hirose, Study on cooperative positioning system: optimum moving strategies for CPS-III, Proceeding of the 1998 IEEE International Conference in Robotics and Automation, Vol. 4, Leuven, Belgium, May, 1998, pp. 2896-2903.

[32] S. I. Roumeliotis, G. A. Bekey, Distributed multi-robot localization, IEEE Transactions on Robotics and Automation, Vol. 18, No. 5, 2002, pp. 781-795.

[33] R. Fierro, P. Song, A. Das and V. Kumar, Cooperative control of robot formation, in: R. Murphey, P. M. Pardalos(Eds), Cooperative control and optimization, Kluwer Academic Publishers, 2002, pp. 73-93.

[34] H. Yamaguchi, J. W. Burdick, Asymptotic stabilization of multiple nonholonomic mobile robots forming group formations, Proceedings of IEEE International Conference on Robotics and Automation, Vol. 4, 1998, pp. 3573-3580.

[35] H. Yamaguchi, A cooperative hunting behavior by mobile-robot troops, The International Journal of Robotics Research, Vol. 18, No. 8, September, 1999, pp. 931-940.

[36] M. Dorigo, V. Maniezzo, and A. Colorni, Ant systems: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol 26, No. 1, 1996, pp. 29-41.

[37] H. Sussmann, J. C. Willems, 300 years of optimal control: from the brachystochrone to the maximum principle, IEEE Control Systems Magazine, Vol.17, No.3, June 1997, pp. 32-44.

[38] J. K. Parrish, W. M. Hammer(Eds.), Animal groups in three dimensions, Cambridge University Press, Cambridge, U.K., 1997.

[39] Y. Liu, K. M. Passino and M. Polycarpou, Stability analysis of one-dimensional asynchronous swarms, IEEE Transaction on Automatic control, Vol. 48, No. 10, 2003, pp. 1848-1854.

[40] E. M. Khazen, Searching for optimal trajectory with learning, IEEE Transaction on Systems, Man, and Cybernetics - Part A: Systems and Humans, Vol. 31, No. 6, Nov. 2001.

[41] N. E. Leonard and E. Fiorelli, Virtual leaders, artificial potentials and coordinated control of groups, Proceedings of the 40th IEEE Conference on Decision and Control, Orlando, Florida, December, 2001, pp. 2968-2973.

[42] P. Ögren, E. Fiorelli and N. E. Leonard, Formation with a Mission: Stable Coordination of Vehicle Group Maneuvers, Proceedings Fifteenth International Symposium on Mathematical Theory of Networks and Systems, Notre Dame, IL, August, 2002.

[43] P. Ögren, E. Fiorelli and N. E. Leonard, Cooperative control of Mobile Sensor Networks: Adaptive gradient climbing in a distributed environment, submitted to IEEE Transactions on Automatic Control.