Optimization via Local Pursuit: Experiments and Simulations

D. Hristu-Varsakelis [a,*]

[a] *Department of Applied Informatics, University of Macedonia, Thessaloniki 54006, Greece*

August 26, 2006

Abstract

Local pursuit is a cooperative optimal control strategy which mimics the process by which ants gradually optimize their foraging trails. It requires only short-range, limited interactions between members of a group of control systems and converges to an optimal solution by iteratively optimizing an initial feasible trajectory/control pair. With local pursuit, group members solve an optimal control problem in "small" pieces, using only information local to them. This report is a compilation of experiments and simulation examples designed to illustrate the performance of local pursuit. It includes references to articles which discuss the method in detail.

Key words: Co-operative control, Optimization, Agents, Minimum-time control

1 Introduction

Local pursuit is a cooperative, decentralized algorithm for learning optimal controls/trajectories, starting from a feasible solution. Under local pursuit, an ordered sequence of agents (physical or simulated copies of a control system) proceed by calculating (and evolving along) optimal trajectories from their own state to that of their predecessor. Neighboring agents are thus termed "leader" and "follower". Local pursuit mimics the way in which ant colonies optimize their foraging trails and solves an optimal control problem in many "short pieces"; it offers storage and information advantages compared to when an optimal control problem is solved all at once or by a single agents.

This report contains a series of examples illustrating the application of local pursuit. It is intended as an addendum to a series of recent publications on cooperative control. For a complete description of local pursuit (including a mathematical definition and relevant convergence proofs), see the following articles: [1] introduces the term "local pursuit" and contains an early discussion of that method in $\mathbb{R}^2$; [4] extends the discussion to spaces with curvature; [6,7,13] contain a rigorous discussion of a local pursuit algorithm, where agents continuously adjust their trajectories; [11] discusses a "sampled" version of local pursuit which requires only periodic updates to be made to agents' trajectories; finally, [13,5] present an extended version of local pursuit which can solve optimal control problems with partially-constrained final state.

2 Simulations and Experiments

This section describes a series of simulations and an experiment designed to illustrate the performance of local pursuit. All simulation code was written in Matlab and ran on a 1.6GHz Pentium-M PC. In the cases where there was no analytical solutions available for the follower-to-leader optimal trajectories, local pursuit required an existing numerical optimization algorithm as a "subroutine", so that each agent can compute optimal trajectories to its leader [5]. Here, we used the RIOTS [2] optimization software for this purpose, although other numerical optimization codes could be used just as easily. See [5,12] for additional details on the numerical performance of local pursuit.

2.1 A trail optimization problem with free final states

Consider the problem of finding shortest paths in an environment consisting of a plane with two right cones, whose (partial) top view was shown in Fig. 1. The radii of the cones were 800 and 1000 units of length, respectively.

* Tel: +30-2310-891721, Fax: +30-2310-891-290.
Email address: dcv@uom.gr (D. Hristu-Varsakelis).

Each object (the plane and each cone) was parametrized with its own set of coordinate functions. The agents were governed by $\dot{x}_k = u_k, \|u_k\| = 1$ and were required to travel from $x_I = (3500, 0, 0)$ to the second cone.

Fig. 1 shows the iterated trajectories generated when the agents implemented the local pursuit policy (continuous version from [6]) with $T_0 = 3499,\ \Delta = 0.2T_0$. For the computation of the optimal trajectory, each agent had to solve its own optimal control problem which was simpler than the "global" problem, partly because of the fact that the globally optimal trajectory crosses multiple coordinate patches from the plane to the cone(s) and vise versa. When the leader and follower were both on the plane, or on the same cone, the computation of optimal trajectories was straightforward. In other cases, agents had to optimize trajectories that crossed at most two coordinate patches (plane-to-cone or cone-to-plane), selecting from a one-parameter family of curves joining leader and follower. On the other hand, computing the globally optimal trajectory at once would have required searching over a four-parameter family of curves (there are a total of four "crossings" between coordinate patches). A detailed accounting of the computational requirements and numerical performance of local pursuit can be found in [12].

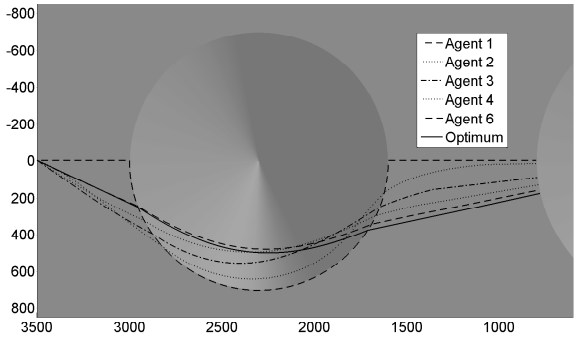


Fig. 1. Continuous local pursuit in a complex environment. The initial trajectory (along the borders of the cones) is easily described but far away from optimal. The locally optimal trajectories were easier to compute than the global optimum because of the limited pursuit distance ($\Delta = 0.2T_0$). The iterated trajectories converged to the optimum.

2.2 Pursuit with a trio of indoor robots

(From [4]). We performed an experiment designed to illustrate local pursuit using the robots shown in Fig. 2. Each robot has three wheels, two of which are independently actuated. The wheel configuration makes the robot kinematically equivalent to a unicycle. The robots were outfitted with sonar and odometry sensors, and had wireless access to the Internet. Their top speed is $2m/s$ and their sensors can be polled at a rate of $30Hz$. In addition, each robot is outfitted with a pair of microphones and speakers. This arrangement allows robots to exchange sound data and get bearing information on one another over short distances.

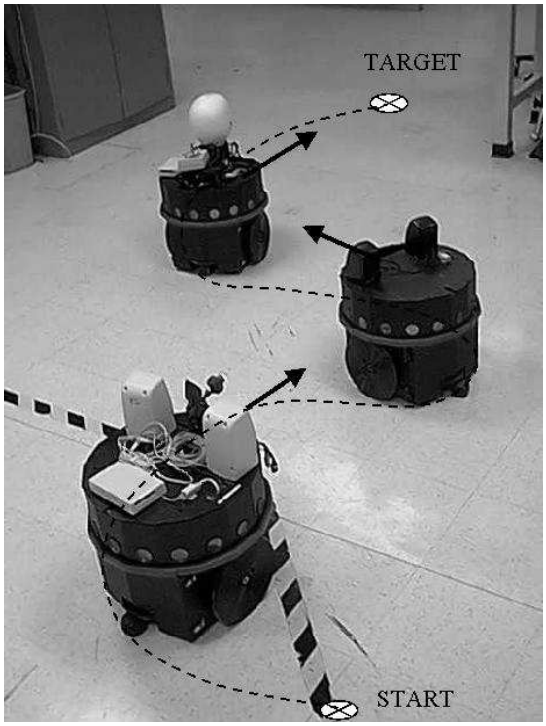
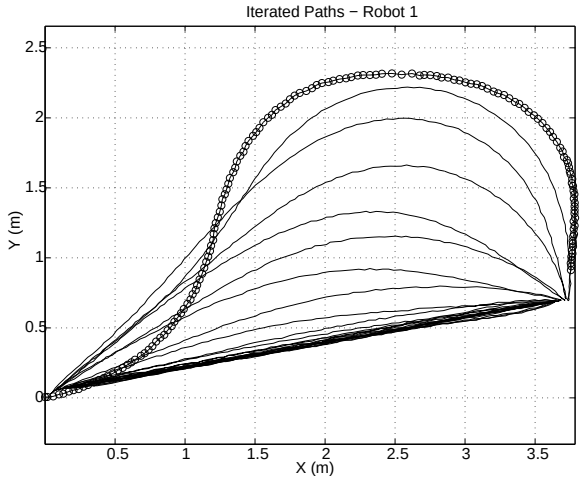
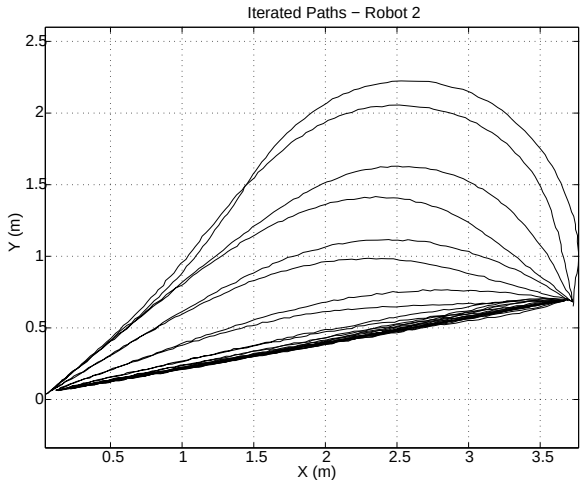


Fig. 2. Local pursuit with a trio of robots.

The robots were designed for indoor use, therefore the experiment described below was performed on level terrain. We fixed a coordinate frame in the room where the robots were located. Starting at the origin, one of the robots (designated as the leader) was sent out to explore the terrain, recording its odometry data along the way and using its sensors to avoid collisions with obstacles. The leader reached the coordinates $(3.75m, 0.75m)$ which were designated as the target, and returned to the origin by following (backwards) the odometry information it collected on its way to the target. Once back at the origin, the leader turned around and re-traced its original path to the target, this time followed by the two other robots, each separated $0.5m$ from the next. We measured length on the plane using the usual Euclidean metric. Each robot followed its leader by moving forward with constant speed, while adjusting its turn rate so as to keep the leader directly ahead. As predicted, each follower robot traveled less distance than its leader, effectively shortening the path between the origin and the target. Once at the target location, the robots followed each other back to the origin, further reducing the



(a)



(b)

Fig. 3. Iterated paths created by "following": (a) first and (b) second robots.

total length traveled. We arranged matters so that the robots followed one another back and forth between the origin and the target, in order to circumvent the need for a large number of vehicles, Figure 3 shows the paths traveled by the first (leader) and second robots during seven successive trips between the origin and the target. The curve highlighted with small circles indicates the initial path. As expected, the iterated paths approached a straight line.

It should be mentioned that by choosing R^2 with the Euclidean metric we have effectively ignored the nonholonomic constraint which governs the kinematics of our robots. We were able to do this because following did not require the robots to move sideways. Of course, a more natural choice would have been to regard the robots' configuration space as being $SE(2)$ with an appropriate choice of metric and to look for geodesics in that space. This would involve solving a rather cumbersome two-point boundary value problem on-line, in order to compute the geodesics on $SE(2)$.

2.3 Minimum-time control with speed and acceleration constraints

Consider the minimum-time control of the second-order system

$$\ddot{x} = u; \quad s.t. \ |u| \leq 30, \ |\dot{x}| \leq 8$$

where we seek to minimize $J(x, \dot{x}, 0) = T$, with boundary conditions $\dot{x}(0) = \dot{x}(T) = x(0) = 0$, and $x(T)$ fixed. The optimal policy for this problem is an instance of the well-known 'bang-off-bang" control: u switches at most once between 30 and -30, with $u = 0$ when the maximum for $|\dot{x}|$ has been reached. The initial, suboptimal input (Agent 1 in Fig. 4), alternated between the maximum and minimum available acceleration. When using continuous local pursuit with $\Delta = 1.3$ sec, the third agent's trajectory was optimal (see Fig. 4). Notice that after $t > 2.7$ sec the second agent intercepted the first and subsequently moved along the same trajectory, x_1. It is also interesting to note that in this case, optimality was achieved after a finite number of iterations.

2.4 An experiment in minimum-time control

We implemented the example of Sec. 2.3 using a collection of three motors, pictured in Fig. 5. Each motor was equipped with position and speed sensors, which were sampled by a PC-based controller at a rate of $2000Hz$. The goal was to rotate the motors to a fixed final position in minimum time. Motor acceleration and speed were limited to 30 rad/sec^2 and 8 rad/sec, respectively. The input to the first motor was a rectangular pulse with amplitude equal to the maximum acceleration (same as in the simulation of Sec. 2.3). Each of the remaining two motors tried to "catch up" with its predecessor by measuring the predecessor's state and applying a control to reach that state in minimum time.

The trajectories of all three motors with $\Delta = 1.3$ sec are shown in Fig. 6. We see that the third motor evolved under essentially optimal control, and the second motor "intercepted" the first after $t \approx 2.3$ sec. We note that because of unmodeled friction, the final position $\theta(T)$ was less than the nominal value (see $x(T)$ in the last simulation). The presence of friction also caused the motors to decelerate when a zero input was applied (once the motors had reached maximum speed). That deceleration in turn caused the local pursuit policy to try and "catch up" by introducing a positive control input, resulting in chatter observed in the velocity and acceleration curves of motors 2 and 3 in Fig. 6.

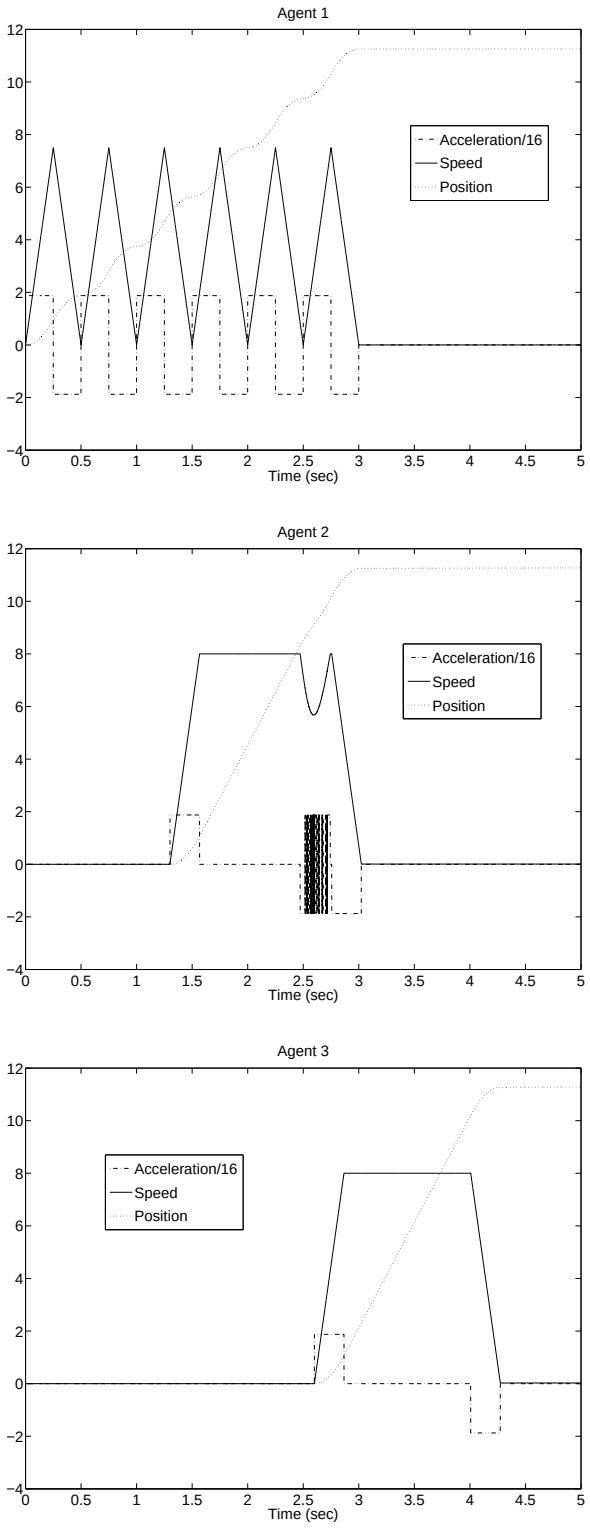


Fig. 4. Iterative trajectories for minimum control with limited acceleration and speed. The pursuit interval was $\Delta = 1.3$. Units for acceleration, velocity and position are $rad/s^2, rad/s, rad$, respectively.

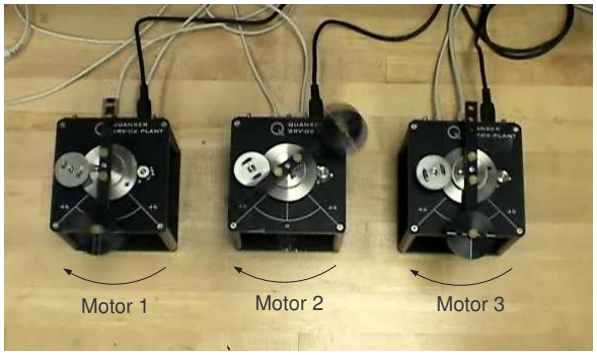


Fig. 5. Applying local pursuit with a trio of motors to obtain minimum-time control with limited acceleration and speed.

2.5 Optimal control of a container crane

Next, we simulated a series of agents whose dynamics were given by the container crane model from [14,10]:

$$\begin{aligned} &\dot{x}_1 = x_4, \quad \dot{x}_2 = x_5, \quad \dot{x}_3 = x_6, \\ &\dot{x}_4 = u_1 + 17.2656x_3, \quad \dot{x}_5 = u_2, \\ &\dot{x}_6 = -(x_1 + 27.0756x_3 + 2x_5x_6)/x_2, \end{aligned} \tag{1}$$

subject to $|u_1(t)| \leq 2.83374$, $-0.80865 \leq u_2(t) \leq 0.71265$, and state constraints $|x_4(t)| \leq 2.5$, $|x_5(t)| \leq 1.0$, where by slight abuse of notation we have used $x_1, ..., x_6$, to indicate the components of the state vector. The initial state was $x_I = [0, 22, 0, 0, -1, 0]^T$. The motion to be optimized was similar to that in [14], where the crane is initially moving downwards and transitions to a state $x_f = [4,\ 19,\ 0,\ 2,\ 0,\ 0]$, at $t = 9$s, where it has been translated and moving horizontally. The cost to be minimized was $J = \int_0^9 x_3^2 + x_6^2 dt$. To generate the initial feasible trajectory, the first agent moved optimally to an intermediate state $x_m = [0,\ 19,\ 0,\ 0,\ 0,\ 0]^T$ (a downward movement to a stop) at $t = 4.5$s, and then optimally again to the desired final state, x_f, at $t = 9$s, at a total cost of 0.02149. Subsequent agents applied local pursuit ("sampled" version of the policy; see [13,5]) with $\delta = 0.5$, $\Delta = 4$, in order to find the overall optimal control/trajectory pair from x_I to x_f. In this case the optimal control problem had no known analytical solution; each follower computed the optimal control to its leader numerically by calling RIOTS [2], and applied that control for δ time units, before repeating the procedure.

With a termination threshold of $\varepsilon = 10^{-6}$, sampled local pursuit performed 29 iterations. Figure 8 shows the iterated costs. The trajectory of the 29th agent (Fig. 9) had a cost of 0.015719, which was within 10^{-5} of the optimum, 0.015714. The optimal trajectory and inputs agreed with those obtained using RIOTS to solve the problem "in one piece". During computation, each follower-to-leader trajectory segment was sampled at 20 knot points. Each iteration of local pursuit (i.e., each agent's trajectory)

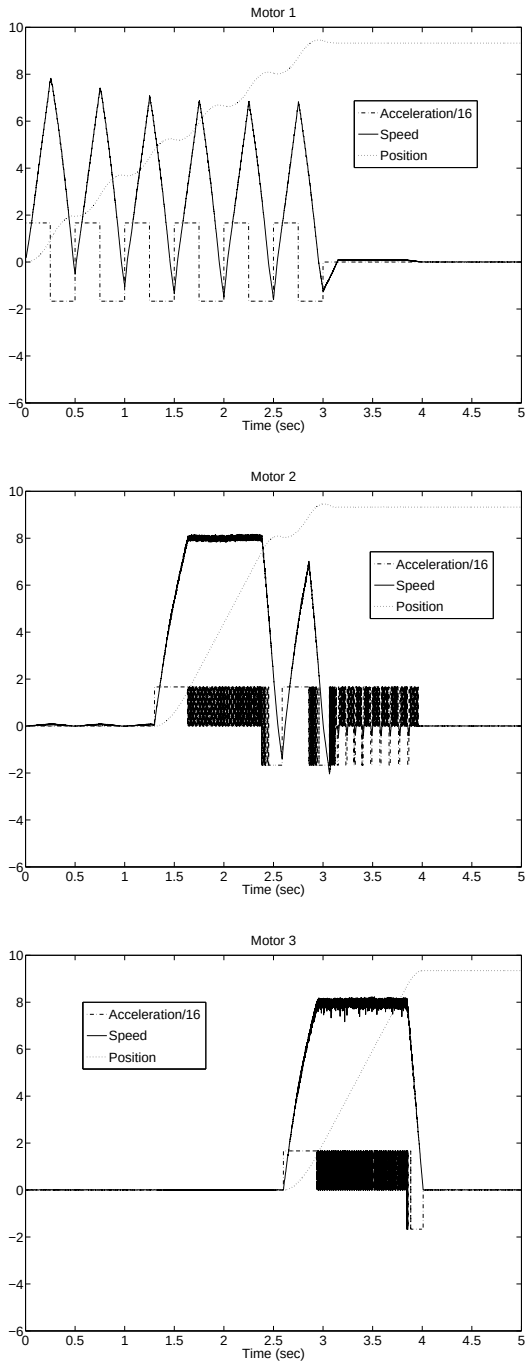


Fig. 6. Iterative trajectories of motors when applying local pursuit to attain minimum-time control with limited acceleration and speed. The pursuit interval $\Delta = 1.3$. The third motor evolved under essentially optimal control.

took an average 450s to compute, and required the corresponding agent to solve 10 versions of the optimal control problem with a time horizon of 3.5s, using approximately 380Kb of memory. This should be compared with 193s and approximately 1840Kb of memory which were required to compute the optimal trajectory all at once,

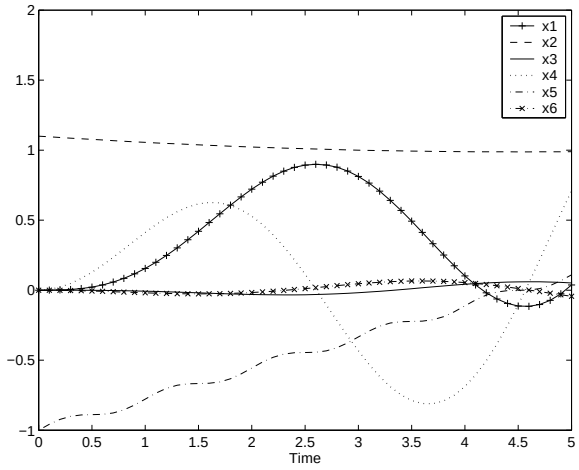


Fig. 7. Initial state trajectory for the container-crane system. The second state is scaled by 1/20.

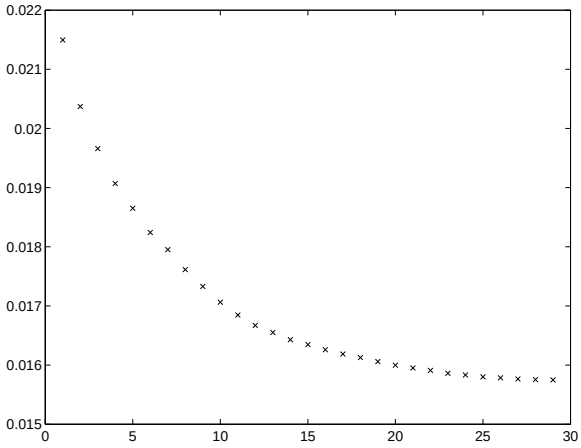


Fig. 8. Cost of each agent's trajectory (container crane).

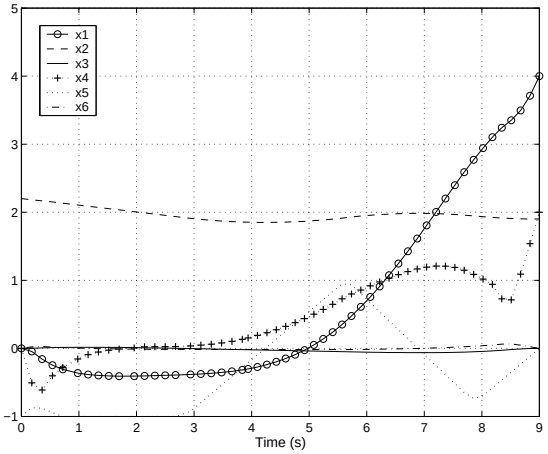


Fig. 9. Trajectory of the 29th agent in the container-crane example. The second state is scaled by (1/20). The trajectory cost was less than 10^{-6} of the minimum.

using the same optimization routine and knot point density.

2.6 Minimum-time control of a bridge crane

Next, we simulated a series of agents whose dynamics were given by the bridge crane system used in [9,8]

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = u, \quad \dot{x}_3 = x_4,$$
$$\dot{x}_4 = -0.98x_3 + 0.1u, \tag{2}$$

with $x_I = 0$ and $|u(t)| \leq 1$. To generate an initial trajectory, we had the first agent proceed to the intermediate state $x_m = [5.0554,\ 0.1862,\ 0.1444,\ -0.0307]^T$ using the control $u(t) = \sin(2\pi t/4)$, $t \in [0, 7.5]$, and then on to the final state $x_f = [15,\ 0,\ 0,\ 0]^T$ optimally from x_m. Subsequent agents applied sampled local pursuit with $\delta = 1$ and $\Delta = 5$ in order to find the minimum-time trajectory from x_I to x_f. We transformed the minimum time problem into a partially-constrained final state problem by defining a new state, x_5, with $\dot{x}_5 = 1$, and minimizing x_5 at the time when the state satisfies $[x_1, ..., x_4]^T = x_f$.

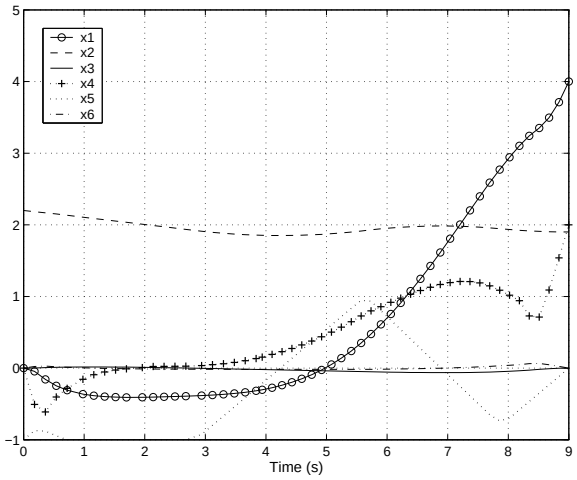


Fig. 10. Trajectory of the 29th agent in the container-crane example. The second state has been scaled by (1/10).

During pursuit, each follower called RIOTS to compute the minimum-time control to its leader, and applied that control for δ time units, before re-adjusting its trajectory. The initial trajectory reached x_f in 14.5748s; subsequent agents arrived in 13.9376s, 12.5651s, 12.0216s, 11.6930s, 11.1640s, 9.3250s, 8.5663s, and 8.5810s. The trajectory of the 10th agent was optimal (8.5808s). The optimal cost, trajectory and control, all agreed with those given by RIOTS when solving the problem in its entirety, as well as with the values reported in [8]. RIOTS used 50 sampling points for each follower-to-leader trajectory segment. Each pursuit iteration took an average 9.81s to complete, and used 128Kb of memory. This should be compared with 2.94s and 488Kb of memory which were required to compute the optimal trajectory all at once using the same optimization routines and trajectory sampling density.

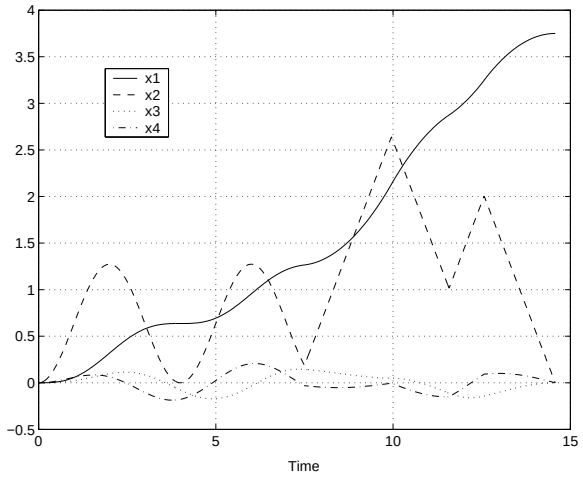


Fig. 11. Initial trajectory for the bridge crane example.

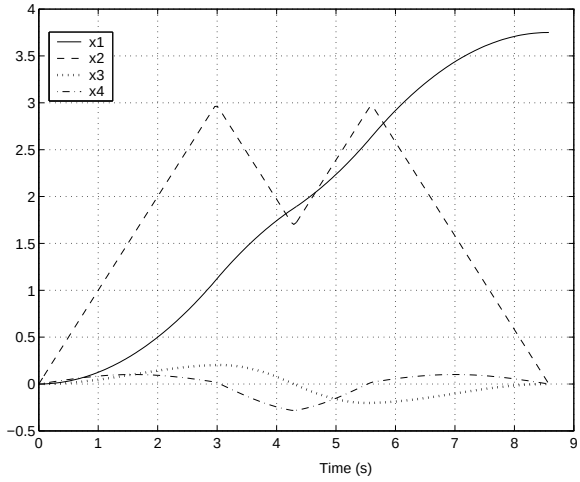


Fig. 12. Trajectory of the 10th agent (optimal) for the bridge crane example. The first state is scaled by (1/4).

2.7 Minimum-time control of a quadruple integrator

Finally, we let the agent dynamics be given by the quadruple integrator [3,8]:

$$\frac{d^4 z}{dt^4} = u, \quad |u(t)| \leq 1, \tag{3}$$

where $x \triangleq [z,\ dz/dt,\ d^2z/dt^2,\ d^3z/dt^3]^T$, and $x_I = [0.1, 0.2, 0.3, 0]^T$. We let the first agent evolve to $x_f = [5.4012,\ 0.4627,\ 0.3,\ 0]^T$ using the control $u(t) = -0.7\cos(2\pi t/4)$, $t \in [0, 16]$. Subsequent agents applied sampled local pursuit [5] with $\delta = 1$ and $\Delta = 3$ in order to find the minimum-time trajectory to x_f.

We transformed the problem into one with partially-constrained final state, by appending an additional fifth state, $\dot{x}_5 = 1$ to (3). We again used RIOTS to compute the minimum-time control from each follower to its leader, and applied that control for δ time units, before repeating the procedure. The second-through-seventh agents reached the target state in 14.5445s, 11.8238s, 11.2727s, 8.7919s, 6.4578s, and 6.1688s. The trajectory of the eighth agent was optimal, with a duration of 6.1687s. As in [8], the optimal control was the well-known four-stage bang-bang policy. Calls to RIOTS used 30 sampling points for each follower-to-leader trajectory segment. Each iteration of local pursuit took an average of 5.84s, and used 54Kb of memory, compared with 3.4s and 396Kb of memory which were required to compute the optimal trajectory all at once using the same routines and sampling density.

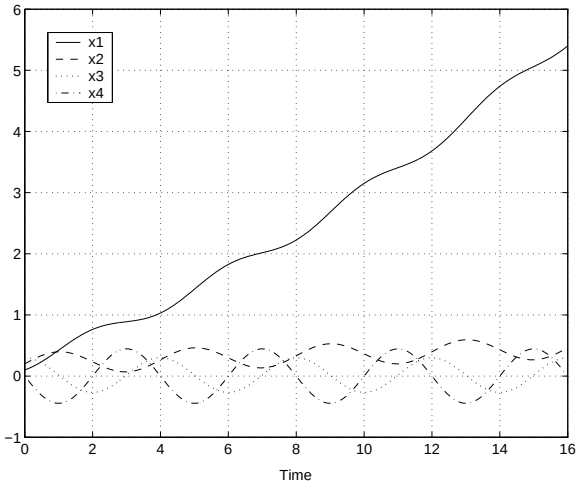


Fig. 13. Initial trajectory for the quadruple integrator example.

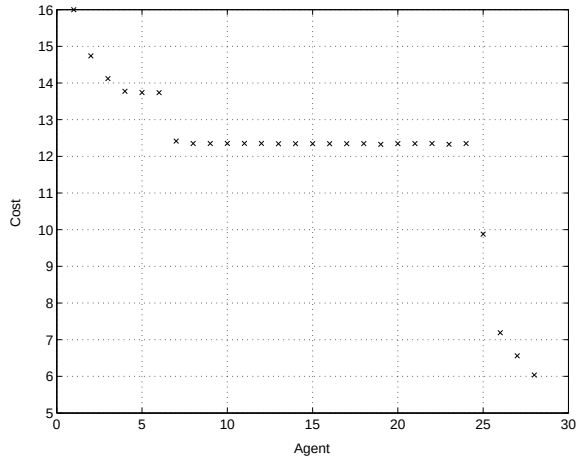


Fig. 14. Iterated trajectory costs for the quadruple integrator example.

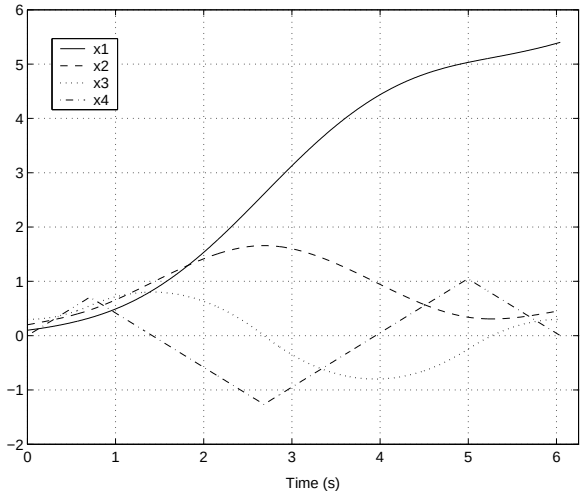


Fig. 15. Trajectory of the 8th agent (optimal) for the quadruple integrator example.

3 Acknowledgments

The authors would like to thank Prof. Y. Chen and Dr. A. Schwartz for providing a copy of the RIOTS software, as well as Prof. R. Luus for helpful discussions on the examples of Sec. 2. The authors also thank the anonymous reviewers for their detailed and helpful feedback.

References

[1] A. M. Bruckstein. Why the ant trails look so straight and nice. *The Mathematical Intelligencer*, 15(2):59–62, 1993.

[2] Y. Chen and A. L. Schwartz. Riots95 - a Matlab toolbox for solving general optimal control problems and its applications to chemical processes. In R. Luus, editor, *Recent developments in optimization and optimal control in chemical engineering*, pages 229–252. Transworld Research Pub., 2002.

[3] T. S. Chung and C. J. Wu. A computationally efficient numerical algorithm, for the minimum-time control problem of continuous systems. *Automatica*, 72:841–7, 1992.

[4] D. Hristu-Varsakelis. Robot formations: Learning minimum-length paths on uneven terrain. In *Proc. 8th IEEE Mediterranean Conf. on Control and Automation*, 2000.

[5] D. Hristu-Varsakelis and C. Shao. A bio-inspired pursuit strategy for optimal control with partially-constrained final state. *Automatica*. prov. accepted.

[6] D. Hristu-Varsakelis and C. Shao. Biologically-inspired optimal control: Learning from social insects. *Int'l Journal of Control*, 77(18):1545–66, Dec. 2004.

[7] D. Hristu-Varsakelis and C. Shao. Corrections to: Biologically-inspired optimal control: Learning from social insects. *Int'l Journal of Control*, 78(2):157, Jan. 2005.

[8] R. Luus. *Iterative Dynamic Programming*. Monographs and Surveys in Pure and Applied Mathematics. Chapman & Hall/CRC, 2000.

[9] S. M. Moon and H. F. VanLandingham. On the variational approach in a time optimal control problem. In *Proc. IASTED Int'l Conf. CONTROL'97*, pages 290–3, 1997.

[10] Y. Sakawa and Y. Shindo. Optimal control of container cranes. *Automatica*, 18:257–266, 1982.

[11] C. Shao and D. Hristu-Varsakelis. Bio-inspired optimal control via intermittent cooperation. In *Proc. of the American Control Conference*, pages 1060–65, Jun. 2005.

[12] C. Shao and D. Hristu-Varsakelis. Local pursuit as a bio-inspired computational optimal control tool. Technical Report 2005-85, ISR, University of Maryland, College Park, MD., 2005. Available online at: http://techreports.isr.umd.edu/reports/2005/TR_2005-85.pdf.

[13] C. Shao and D. Hristu-Varsakelis. Optimal control through biologically inspired pursuit. In *Proc. of the 16th IFAC World Congress*, Prague, Jul. 2005. in press.

[14] K. L. Teo, L. S. Jennings, H. W. J. Lee, and V. Rehbock. The control parameterization enhancing transform for constrained optimal control problems. *J. Austral. Math. Soc. Ser. B*, 40:314–335, 1999.