Fall in! Sorting a group of robots with a continuous controller

Yaroslav Litus, Richard T. Vaughan
Autonomy Lab, School of Computing Science, Simon Fraser University, Canada
{ylitus, vaughan}@sfu.ca

Abstract

This paper describes the first robotic system that solves a combinatorial computational problem by means of its own continuous dynamics. The goal of the system is to rearrange a set of robots on a line in a certain predefined order, thereby sorting them. Conventional pairwise between-robot rank comparisons suggested by traditional discrete state sorting algorithms are avoided by coupling robots in a Brockett double bracket flow system. A conventional multi-robot simulation with non-holonomic driving, noisy sensor data, collision avoidance and sensor occlusions suggests that this flow system can withstand perturbations introduced into the ideal dynamics by the physical limitations of real robots.

1 Introduction

Ordering robots may be a necessary part of many robotic tasks. For example, a team of robots may need to board a transport vehicle in a certain order to use cargo space more efficiently. Likewise, a certain order may be preferred when deploying robots from that vehicle. Maintaining a priority queue of robots may be required when robots line up for service or refueling (see Fig. 1) or when robots perform a convoying task. Finally, ordering robots may improve performance of certain spatial interference resolution algorithms [17].

Sorting is usually solved by algorithms running on discrete state machines. To our knowledge this is the first paper to explicitly consider performing such a discrete computation by a continuous dynamics of a multi-robot system. Other multi-robot systems either use continuous dynamics to solve continuous problems [6,8] or use discrete dynamics to solve discrete problems [10]. We describe a decentralized multi-robot controller that sorts robots by coupling them according to the Brockett double bracket flow system. This controller is a novel robotic application of the well-known Brockett system.

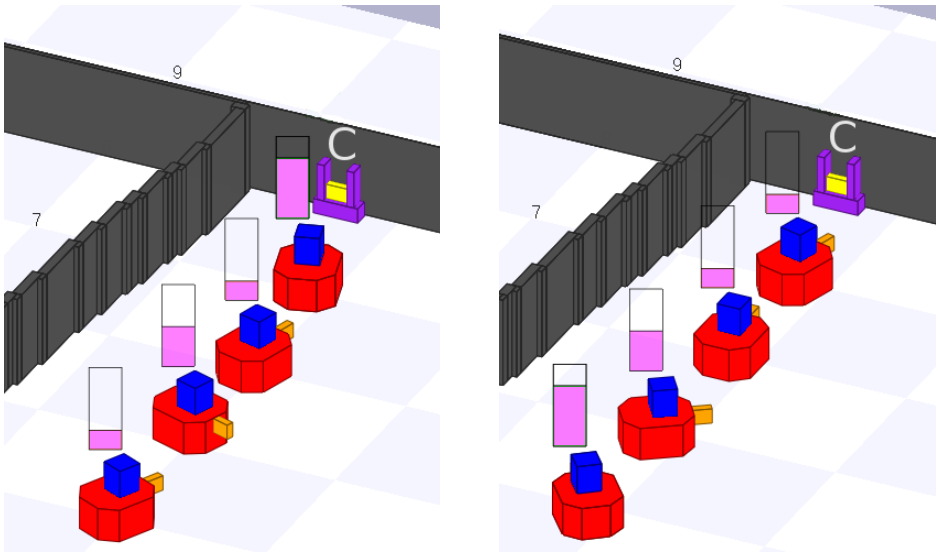
Formally, let n robots on a plane be initially positioned on a line $y = y_0$ at coordinates $(x_i(0), y_0)),\ i = 1, \ldots, n$ in a Cartesian coordinate system. Assuming, without loss of generality, that the sought ordering coincides with the robot numbering, the goal is to drive the system to a state $(x(t), y(t))$ where $x_1(t) < x_2(t) < \ldots < x_n(t)$ and $y_1 = y_2 = \ldots y_n = y_0$. That is, we want the robots to sort themselves along the horizontal axis and return to the line where they started. The sorting computation should be performed solely by the dynamics of interacting and moving robots.

The next section presents related work, which is followed by a description of the Brockett sorter, a dynamical system capable of performing sorting by means of smooth dynamics. After an informal theoretical argument about issues that may arise in using the Brockett sorter in a real robotic system we describe a controller based on this sorter. The controller is tested in a short demonstration followed by discussion of the observed behavior and properties of the system. The paper concludes by suggesting several directions for future work.

2 Related work

The idea of viewing robotic systems as computational devices is related to the Brooks' advice against internalizing the world [5]. As long as the computation problem faced by a robotic system is set in terms of the variables that the system can modify through its behavior, required computation can potentially be performed by that behavior. The computation is hence externalized and happens outside of the internal information processing unit. From this point of view a robotic system can perform computations not only by using its conventional dedicated processing units, but also by its sensors and actuators.

These computational capabilities have some recognition in the robotics community. In the domain of single robot systems Paul [9] and Pfeifer [11] show that robot morphology can assume a computational role. In multi-robot systems Payton et.al. [10] introduce a concept of "world embedded computation" and describe a robot swarm system that runs an analogue of Dijkstra's shortest path algorithm



(a) Robots lined up in arbitrary order. The last robot in the queue has a high chance of running out of charge before reaching the station.

(b) Robots are sorted by the remaining charge. The chance of starvation is minimized.

Figure 1. Robots queuing for recharging at the station "C" located near the wall. The share of remaining charge is shown for every robot. Sorting robots increases the probability that all robots can charge before running out of energy.

in a world embedded manner. Robots spread themselves in the environment and then find the shortest path between two points by exchanging local messages. Hamann and Wörn [6] discuss the idea of embodied computation and present a swarm system that computes an approximation to the solution of geometric Steiner tree problem. Litus and Vaughan [8] argue that embodiment and spatial embeddedness can serve as a surrogate for computational resources for developing decentralized distributed gradient descent optimization algorithms for teams of embodied agents.

Since robotic systems evolve in continuous state space, the relation between discrete computation and continuous systems is very important for treating robot systems as computers. Brockett [4] shows that double-bracket matrix flow dynamical systems can serve as an analog computer solving a variety of combinatorial problems including sorting. These systems bear similarity to finite aperiodic Toda lattices, a recent thorough review of which is given by Kodama and Shipman [7]. Saxena and Clark [12] describe an electronic implementation of Brockett double bracket flow built from analogue integrated circuits. Zavlanos and Pappas describe a dynamical system inspired by the double bracket flows that approximates the solution to the combinatorial weighted graph matching problem [15]. Bloch and Crouch [3] argue that from control theoretical point of view some combinatorial problems can be seen as minimization of a function over a set of configurations which can be contrasted with minimization over a class of curves in classical optimal control. The problem of sorting robots could be related to formation control (see Bahceci et.al. [1] for a recent review).

3 Brockett smooth sorter

In a seminal paper [4] Brockett analyzes dynamical systems

$$\dot{H} = [H, [H, N]] \tag{1}$$

where H and N are symmetric matrices and $[A, B] = AB - BA$. He shows that these so called "double bracket flow" systems define an isospectral gradient flow, so as H evolves its eigenvalues do not change. The author proves that these systems can serve as a versatile analog computer solving linear programming problems, certain combinatorial optimization problems and diagonalizing symmetric matrices. This last ability is of a particular interest for robot sorting since it provides means to sort a list of numbers by a smooth dynamical system.

Brockett proves that if N is a diagonal matrix with distinct elements, the equilibrium matrix $H(\infty)$ will be a diagonal matrix with its elements arranged in the same order as elements of N. In particular, if $N = \mathrm{diag}(1, 2, \ldots, n)$ then for almost all Θ and for

$$H(0) = \Theta^T(\mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n))\Theta$$

the equilibrium $\dot{H} = [H, [H, N]]$ will approach

$$H(\infty) = \text{diag}(\lambda_{\pi(1)}, \lambda_{\pi(2)}, \ldots, \lambda_{\pi(n)})$$

where permutation π sorts the final list by its size. Our choice of $H(0)$ is dictated by a need to decrease the number of state variables in order to simplify the controller. Diagonal $H(0)$ has the fewest number of variables, but produces no dynamics. However, symmetric tridiagonal matrix $H(0)$ will produce the desired dynamics and result in $H(t)$ being symmetric tridiagonal for any time t. Setting

$$H(0) = \begin{pmatrix} b_1 & a_1 & & \\ a_1 & \ddots & \ddots & \\ & \ddots & \ddots & a_{n-1} \\ & & a_{n-1} & b_n \end{pmatrix} \quad (2)$$

where $b_i, i = 1, 2, \ldots, n$ are the values to be sorted and $a_i, i = 1, 2, \ldots, n-1$ are small non-zero values will make $H(\infty)$ a diagonal matrix containing entries that approach the sorted list of eigenvalues that are close to b_i. The smaller the values of a_i, the closer matrix $H(0)$ is to being diagonal and having b_i as its eigenvalues, and the closer the diagonal of $H(\infty)$ is to the list of sorted components of b_i. Decreasing a_i, though, comes at the cost of increasing convergence time.

System (1) with $N = \text{diag}(1, 2, \ldots, n)$ and $H(0)$ as in (2) is equivalent to symmetric tridiagonal Toda equations [2]. These equations describe the behavior of the system of n unit mass particles arranged along a line with adjacent particles interacting with a magnitude that exponentially depends on the distance between them [13]. Component-wise in terms of a and b dynamics of this system can be written as

$$\dot{a}_k = a_k(b_{k+1} - b_k), \; k = 1, 2, \ldots, n-1 \quad (3)$$

$$\dot{b}_k = 2(a_k^2 - a_{k-1}^2), \; k = 1, 2, \ldots, n \quad (4)$$

with initial conditions $a_0 = 0$.

Fig. 2 illustrates the evolution of the system (3-4) with initial values $b(0) = (3, 1, 6, 2)$, $a(0) = (.001, .001, .001, .001)$. The system was simulated in discrete time as $b(t+1) = b(t) + \delta\dot{b}(t)$, $a(t+1) = a(t) + \delta\dot{a}(t)$ with discretization parameter $\delta = 0.005$. Four lines on the figure show the values of four components of vector b plotted against simulation time. After 3500 steps the value of the vector b converges to $b(3500) = (6.088, 2.976, 1.999, 0.937) \approx (6, 3, 2, 1)$: the desired ordering.

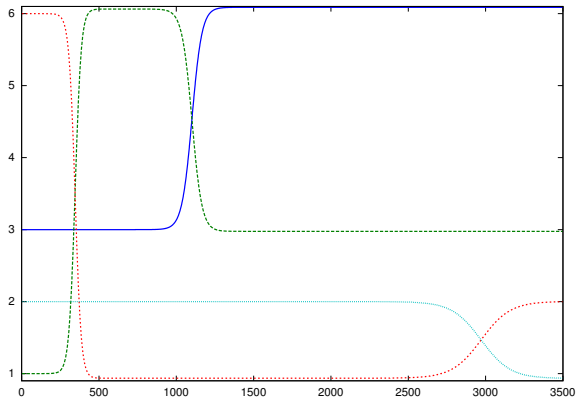


Figure 2. Dynamics of sorting system (3-4) initialized with $b(0) = (3, 1, 6, 2)$, $a(0) = (.001, .001, .001, .001)$. Four components of vector b (vertical axis) are plotted against time (horizontal axis). In 3 500‘ simulation steps the system converges to $b = (6.088, 2.976, 1.999, 0.937)$.

4 Application to robot sorting

4.1 Theoretical considerations

The dynamical system (3-4) (hereinafter “the sorting system”) can serve as a basis for a multi-robot system controller that sorts robots. To do this we need to find a way to embed the sorting system into the state space of a multi-robot system. This way the movement of the robots can be dictated by the evolution of sorting system and result in robots sorting themselves.

Regardless of the means of embedding the sorting system into the state space of a multi-robot system we need to acknowledge the limitations a realistic robot system imposes on the possible trajectories in a state space. For example, robots have limited speed and acceleration and may be non-holonomic. Also, a real system will have noise present in sensor readings and responses to control inputs. Finally, collision avoidance should be employed by a real robot system introducing additional constraints on trajectories. This raises an important question: is the sorting system able to withstand some perturbation of the state variables as the system evolves and still converge to the same equilibrium? A brief analysis shows that this is not the case. Assume that sorting system evolves from tridiagonal matrix $H(0)$ to diagonal matrix $H(\infty) = \text{diag}(\lambda_{\pi(1)}, \lambda_{\pi(2)}, \ldots, \lambda_{\pi(n)})$ and at time t one of the state variables was perturbed resulting in matrix $H'(t) \neq H(t)$. As long as the spectrum of $H'(t)$ differs from the spectrum of $H(t)$ the system will now converge to a different equilibrium $H'(\infty) = \text{diag}(\lambda'_{\pi(1)}, \lambda'_{\pi'()}, \ldots, \lambda'_{\pi(n)})$. There is no feedback in the system to correct this deviation from the original trajectory and restore original spectrum of H. In this sense the sorting system is fragile and using it as a base for a robot sorting

controller seems to be difficult.

However, the sorting system should not be discarded because of its fragility. Despite the sensitivity of the equilibrium to the perturbation of state variables, all equilibriums are in fact diagonal matrices with sorted eigenvalues. Thus, if some state variable s_i of the i-th robot in the sought ordering behaves as the i-th diagonal entry of H, the equilibrium values $s_i(\infty)$ will follow the sought ordering irrelevant of the changes in the spectrum of H brought by the deviations from the perfect trajectory. In other words, though the actual values of $s_i(\infty)$ will differ from what they could have been in the absence of perturbations, as long as the system is allowed to converge the values of robots state variables $s_i(\infty)$ will be sorted. In this sense the sorting system is reliable and we can attempt to use it to control an appropriately constructed robot system.

4.2 Implementation

We will embed the sorting system into the robot system as follows. x_i will correspond to the diagonal entries of H while $y'_i = y_i - y_0 + \epsilon$ where ϵ is a small non-zero value will correspond to non-diagonal entries of H. In these variables, sorting system (3-4) can be rewritten as a first order controller

$$\dot{x}_1 = 2y_1^2, \tag{5}$$

$$\dot{x}_i = 2(y_i^{'2} - y_{i-1}^{'2}),\ i = 2, \ldots, n-1 \tag{6}$$

$$\dot{y}_i = \dot{y}'_i = -(x_i - x_{i+1})y'_i,\ i = 1, \ldots, n-1 \tag{7}$$

$$\dot{x}_n = -2y_{n-1}^{'2}, \tag{8}$$

$$\dot{y}_n = y'_n = 0; \tag{9}$$

This controller requires every robot i to use the following information:

1. y'_i, the vertical distance from the original line $y = y_0$
2. y'_{i-1}, the vertical distance of the previous robot (if there is one) in ordering from the original line $y = y_0$
3. $(x_i - x_{i+1})$, horizontal distance to the next robot (if there is one) in the ordering

Note, that y'_{i-1} can be computed from y'_i if robot i knows the vertical distance to robot $i-1$. Therefore, the controller requires the robot to be partially localized (know an estimate of its y coordinate) and be able to estimate horizontal distance to one robot and vertical distance to another robot. These requirements can be met by various sensory/communication solutions. To avoid any sort of communication, in the demonstration robots are equipped with fiducial detectors that can determine the relative bearing and distance to the previous and next robot in the ordering. Robots are also localized, though we use only orientation and the y coordinate. Horizontal or vertical distance to a team member is calculated from the position of self, and estimates of distance and bearing to the team member. Thus, all information required by the controller is available in this robot system.

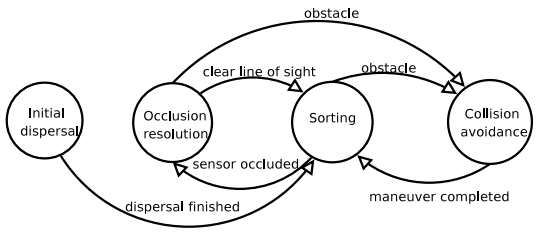


Figure 3. State diagram of the robot sorting controller

There are several issues that emerge when using this controller in a realistic system (or realistic simulation). First, fiducial finders require clear line of sight and hence occlusion can prevent the robot from getting information about distances to other robots. Second, as all sensors, fiducial finders are noisy, therefore information about bearing and distance to the teammates is imprecise. Third, all robots have bounded magnitude of their speed vector, and nonholonomic robots have also bounds on the direction of this vector. Finally, robots can not drive through each other so collision avoidance should be used. All these issues will force the robots to deviate from the trajectory prescribed by the sorting system, however as we argued above we still expect the robots to converge to the sorted order. We will address aforementioned issues in the following paragraphs. The state machine of the resulting controller is shown on Fig. 3.

Occlusions Robots have no knowledge of the team size or their absolute position in the ordering. They are capable of sensing positions of the previous and the next robot in the ordering if the line of sight is not occluded by other robots. Initially every robot assumes that it is the only robot in the team. Once robot i senses the previous robot $i-1$ in the ordering, without sensing robot $i+1$ before, robot i assumes that i is the last robot. It will behave accordingly and use the relative position of robot $i-1$ in its future calculations of speed vector. If robot i senses the next robot $i+1$ in the ordering without previously sensing $i-1$, robot i assumes that i is the first robot. If robot i senses $i-1$ and $i+1$ simultaneously or in any order, it knows that i is not the first and not the last robot and needs relative positions of both $i+1$ and $i-1$ to calculate its speed vector. Therefore, the robot discovers its position in the ordering (last,first, in between) as the dynamics unfolds.

Regardless of the currently assumed position of robot i if its fiducial sensor is occluded and robot i can not see one or both of the robots it needs to calculate the speed vector, it reduces speed to a certain predefined constant without changing its direction. Once the line of sight is restored, the robot selects the desired speed and direction following Eq. (5). Since initially all robots are located along a line and almost all fiducial finders are occluded, the robots disperse themselves by moving with random speeds along the vertical axis for a fixed predefined time. With high probability this allows most of the robots to observe the required teammates and follow Eq. (5-9).

Noise We will not use any noise reduction techniques and instead treat all noisy sensor readings as true values.

Bounds on a speed vector If the magnitude of the desired speed vector $\vec{v} = (\dot{x}, \dot{y})$ exceeds the maximum speed $V_{\max}$ of the robot, the robot tries to set its speed vector to $\vec{v}_{\text{clamped}} = \frac{\vec{v}}{\|\vec{v}\|} V_{\max}$ thus attempting to go in the desired direction with the maximum speed. In the demonstration below we use non-holonomic steering robots which are driven by a simple negative feedback controller

$$\dot{\theta} = \theta - \angle\vec{v} \tag{10}$$

$$s = \|\vec{v}\| \cos(|\dot{\theta}|), \tag{11}$$

where θ is the robot bearing, $\angle\vec{v}$ is the direction of the desired speed vector in the same coordinate system, s is the driving speed.

Collision avoidance We employ a simple collision avoidance algorithm that uses laser range-finder sensor readings. If there is an obstacle closer than a certain distance d_{stop}, the robot stops. If there is an obstacle which is at closer than a certain distance $d_{\text{avoid}} > d_{\text{stop}}$ then the direction that gave the smallest distance reading is found. If smallest reading came from the direction to the right of the robot bearing, a collision avoidance maneuver with a duration randomly selected in a certain interval is performed. The robot starts to turn left with a fixed turning speed and driving speed. Otherwise, the robot performs a right turn maneuver. If smallest reading came from the left, a right turn maneuver is performed. Once the collision avoidance maneuver is over, the robot continues to set the speed as prescribed by Eq. (5-9).

4.3 Demonstration

For the demonstration we use a team of simulated robots. Robots are placed along a horizontal line and the sorting controller is executed simultaneously on every robot. We perform two kinds of simulations. In the first, "idealistic", simulation the robots are holonomic, with no speed restrictions, no sensor occlusions and no collisions. Therefore, the dynamics of this system are described by Eq. (3-4). In the second, "lifelike" simulation Pioneer robots are simulated in the well-known Stage robot simulator [14]. Simulated Pioneer robots can collide, so they need to use collision avoidance, they have non-holonomic driving, top speed restriction and their fiducial sensors can be occluded by other robots. Sensor noise is simulated by adding uniformly distributed random errors to the true distance and bearing reading of fiducial finder before they are used by a controller. Robots in the Stage simulation use the controller described in Section 4.2. Parameters of the Stage simulation are given in Table 1. All simulations run until the speeds of all robots converge to a near-zero value.

Maximum speed	1.2 m/s
Collision avoidance speed	0.1 m/s
Collision avoidance turning speed	0.5 rad/s
Collision avoidance initiation distance	1 m
Minimum front stopping distance	0.5 m
Collision avoidance duration interval	[1,2]s
Speed during occlusion	0.2 m/s
Initial dispersing duration	1.5s
Fiducial finder bearing accuracy	$\pm$3 degrees
Fiducial finder range accuracy	$\pm$ 15 mm

Table 1. Parameters used in Stage simulation

Figure 4 shows the trajectories produced by the robots for three different initial conditions. For every initial condition two trajectories are shown. The first trajectory is produced by an idealistic simulation, the second by a lifelike Stage simulation. Note that Stage simulations are nondeterministic because of the sensor noise and initial random dispersal and infinitely many different trajectories are possible of which we show only one. The idealistic simulation trajectory, on the other hand, is repeatable and fixed up to the rounding and discretization errors during simulation. Each trajectory is described by two graphs. The first graph plots values of x coordinates of robots against simulation time, the second shows the joint (x, y) trajectories of the robots.

In the idealistic simulations most of the robots initially start moving with small speed with vertical component of the speed vector dominating horizontal component. As robots move away from the start horizontal line their speed grows, the horizontal component of the speed vector increases and the vertical component decreases to the point where the vertical component becomes negative and robots begin to return to the start horizontal line but with different horizontal coordinate. Some of the robots move along the start horizontal line for the part of their trajectory while

the last, n-th robot never leaves the start line moving only horizontally (see Eq. (9)). Depending on the initial conditions robots may depart from and return to the start line several times before the robots converge to the sorted order (see, e.g., robots starting at positions $x = 3$ and $x = 5$ on Fig. 4(e),4(f)). Also, robots may switch their vertical direction before reaching the start line (see, e.g., robot starting at positions $x = 13$ on Fig. 4(e), 4(f) and robot starting at position $x = 8$ on Fig. 4(i), 4(j)). Once the sorted state is reached, robots do not depart from it any more. The final configuration has robots rearranged in the sorted order along the start horizontal line with the set of final horizontal positions having values very close to the set of original horizontal positions. Therefore, the robots not only end up in the sorted order, but they also jointly occupy same horizontal positions that where occupied by the team initially.

In the Stage simulations robots initially disperse themselves by moving for a fixed time with a random speed in a vertical direction. This eliminates some of the occlusion and allows some robots to move in the direction prescribed by the sorting system. After dispersal the robots start moving with increasing speeds which is limited by the top speed of the robot. Occlusions that were not resolved by the initial dispersal are eventually resolved as occluded robots move, slowly creating new lines of sight (see Section 4.2). Some occlusions are resolved by the collision avoidance behavior. Robots move away from the horizontal line with horizontal component of their speed vector increasing and vertical component decreasing until the vertical component changes the sign and robots return to the start line at a different horizontal coordinate. Part of the robot trajectory may include a horizontal segment when robot moves along the start line without departing from it. As in the idealistic case, a robot may return to and depart from the start line several times (see robot starting at position $x = 5$ on Fig. 4(g), 4(h)). Robots can also switch their vertical direction before reaching the start line (see the robot starting at position $x = 8$ on Fig. 4(k), 4(l)). If the robots meet, they initiate collision avoidance which may be repeated several times if their desired trajectories lie close to each other. The occlusion resolution described in Section 4.2 ensures that the robots keep moving even if they can not observe one or both of their neighbors, thus eventually restoring the line of sight. Once the robots reach the sorted order, there are no occlusions and robots can finish convergence by bringing their speeds to zero. The final configuration has robots rearranged in the sorted order along or close to horizontal start line. However, they jointly occupy horizontal positions that differ from those occupied by the team initially.

5 Discussion

Both idealistic and lifelike simulations result in successful sorting of the robots. However, in a lifelike simulation robots end up converging to a set of positions that differs from the original set. That agrees with theoretical considerations stated in Section 4.2 as departures from the perfect trajectory caused by speed limitations, occlusions and collision avoidance change the eigenvalues of matrix $H(t)$ in Eq. (1) and, thus, the set of final horizontal positions. Therefore, the limitations of real robots used in simulations break down the position set preservation property of the ideal sorting system while still reaching the goal of sorting robots.

Another difference between the idealistic and life-like simulation is the convergence time. Life-like simulations take more time to converge due to the speed limitations, time spent on collision avoidance, and time spent moving during the occlusions when trajectory may stray away from the convergence path. For example, it takes approximately 10 seconds for idealistic system to sort robots under conditions shown on Fig. 4(a), while life-like Stage simulation takes about 90 seconds. With more robots (see Fig. 4(e)) the idealistic system still takes 10s to sort robots while the life-like simulation takes approximately 300 seconds to converge. A similar difference is observed under different initial conditions (see Fig. 4(i)) where the idealistic system converges in 7 seconds, while the life-like system sorts robots in 250 seconds. Fiducial sensor range imposes another limit on the practicality of this system as an increase in the team size will lead to an increase in the distance between robots which will eventually exceed the fiducial sensor range.

Formal analysis of the behavior of this system, including convergence properties is desirable but presents difficulties. While including non-holonomic driving and noisy sensors and controls into the formal model of the system seems tractable, sensor occlusions and collision avoidance present a major obstacle. Sensor occlusions result in a non-smooth changes in the robot controls and require analysis that is significantly more complicated than the analysis of the double flow system itself. Likewise, the simple threshold based obstacle avoidance algorithm we employ introduces non-smooth transitions into the system which are further complicated by a random selection of the collision avoidance duration. Even if more tractable deterministic repulsive potentials are used for collision avoidance, convergence analysis of the flow based system is prohibitively challenging [16].

Due to the slow convergence observed in simulations the described system should be viewed as a proof of concept rather than a suggested practical solution. While this system can definitely be used in situations where restrictions on the robot communications and sensing preclude using other

sorting methods, it is desirable to find ways to accelerate the convergence and make extensive experimental evaluation of the modified system before it could be recommended as an engineering recipe.

6 Conclusion

We described a multi-robot system sorting controller based on a smooth Brockett double-bracket flow equation. This controller provides a novel demonstration of computational capabilities of multi-robot systems solving a combinatorial problem by means of continuous dynamics. The controller integrates the Brockett system with nonholonomic steering, simple collision avoidance and sensor occlusion resolution. This strength of this approach is in the fact every robot needs to identify only two or one other robots in the team. No conventional pairwise robot-robot comparisons that standard sorting algorithms suggest are necessary. The robots are reactive agents with information between robots exchanged by means of relative position sensing. Robots have no global knowledge of the system state and very limited memory capacity (memory is used only for dispersal and collision avoidance timers and queue position status). These modest information processing requirements for robots come at a cost of slow convergence and a potential need for long distance sensing which is the weakness of this approach.

Future work includes three main directions. The first direction is performance improvement of the robot sorting controller. This includes devising faster and more reliable collision avoidance strategies, trying to include some feedback mechanisms to correct changes in eigenvalues and thus preserve the set of original horizontal positions, looking for ways to bring down the system convergence time and eliminate the need for long-distance sensing. The second direction is looking for other ways to embed the Brockett sorter into a multi-robot system. While this controller uses the vertical coordinate of the robot as a non-diagonal entry in the H matrix, other modalities may be used. For example, robots can display values of non-diagonal entries by emitting sound, or changing the color or intensity of a display light. Also, more state variables may be involved by considering non-tridiagonal matrices H. Finally, as double-bracket flow systems' computational capabilities include more than sorting, these capabilities should be evaluated in the context of multi-robot systems.

Simulation code

In accordance with the Autonomy Lab's policy on code publication, the source code is made available online at `http://www.sfu.ca/~ylitus/`

`robotSortingSources.zip`,
MD5SUM cb0eb487bf06a0e62b75350f377dcd3f.

References

[1] E. Bahceci, O. Soysal, and E. Sahin. A review: Pattern formation and adaptation in multi-robot systems. Technical Report CMU-RI-TR-03-43, Carnegie Mellon University, 2003.

[2] A. M. Bloch, R. W. Brockett, and T. S. Ratiu. A new formulation of the generalized Toda lattice equations and their fixed point analysis via the momentum map. *Bulletin of the American Mathematical Society*, 23(2):477–485, 1990.

[3] A. M. Bloch and P. E. Crouch. Optimal control, optimization, and analytical mechanics. In *Mathematical control theory*, pages 268–321. Springer-Verlag New York, Inc., New York, NY, USA, 1999.

[4] R. Brockett. Dynamical systems that sort lists, diagonalize matrices and solve linear programming problems. In *Decision and Control, 1988., Proceedings of the 27th IEEE Conference on*, pages 799–803 vol.1, Dec 1988.

[5] R. A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6(1&2):3–15, June 1990.

[6] H. Hamann and H. Wörn. Embodied computation. *Parallel Processing Letters*, 17(3):287 – 298, Sept. 2007.

[7] Y. Kodama and B. Shipman. The finite non-periodic toda lattice: A geometric and topological viewpoint, 2008.

[8] Y. Litus and R. Vaughan. Distributed gradient optimization with embodied approximation. In S. Bullock, J. Noble, R. Watson, and M. A. Bedau, editors, *Proc. Int. Conf. on Simulation and Synthesis of Living Systems*, pages 359–365. MIT Press, Cambridge, MA, 2008.

[9] C. Paul. Morphology and computation. In *Proc. Int. Conf. on Simulation of Adaptive Behavior*, pages 33–38. MIT Press, 2004.

[10] D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee. Pheromone robotics. *Auton. Robots*, 11(3):319–324, 2001.

[11] R. Pfeifer, F. Iida, and G. Gómez. Morphological computation for adaptive behavior and cognition. *International Congress Series*, 1291:22–29, 2006.

[12] N. Saxena and J. Clark. Analogue system for eigenvalue computation and sorting based on an isospectral matrix flow. *Electronics Letters*, 31(1):24–26, Jan 1995.

[13] M. Toda. Vibration of a chain with nonlinear interaction. *Journal of the Physical Society of Japan*, 22(2):431–436, 1967.

[14] R. T. Vaughan. Massively multi-robot simulations in Stage. *Swarm Intelligence*, 2(2-4):189–208, 2008.

[15] M. Zavlanos and G. Pappas. A dynamical systems approach to weighted graph matching. In *Decision and Control, 2006 45th IEEE Conference on*, pages 3492–3497, Dec. 2006.

[16] M. Zavlanos and G. Pappas. Dynamic assignment in distributed motion planning with local coordination. *Robotics, IEEE Transactions on*, 24(1):232–242, Feb. 2008.

[17] M. Zuluaga and R. Vaughan. Reducing spatial interference in robot teams by local-investment aggression. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, Alberta, August 2005.

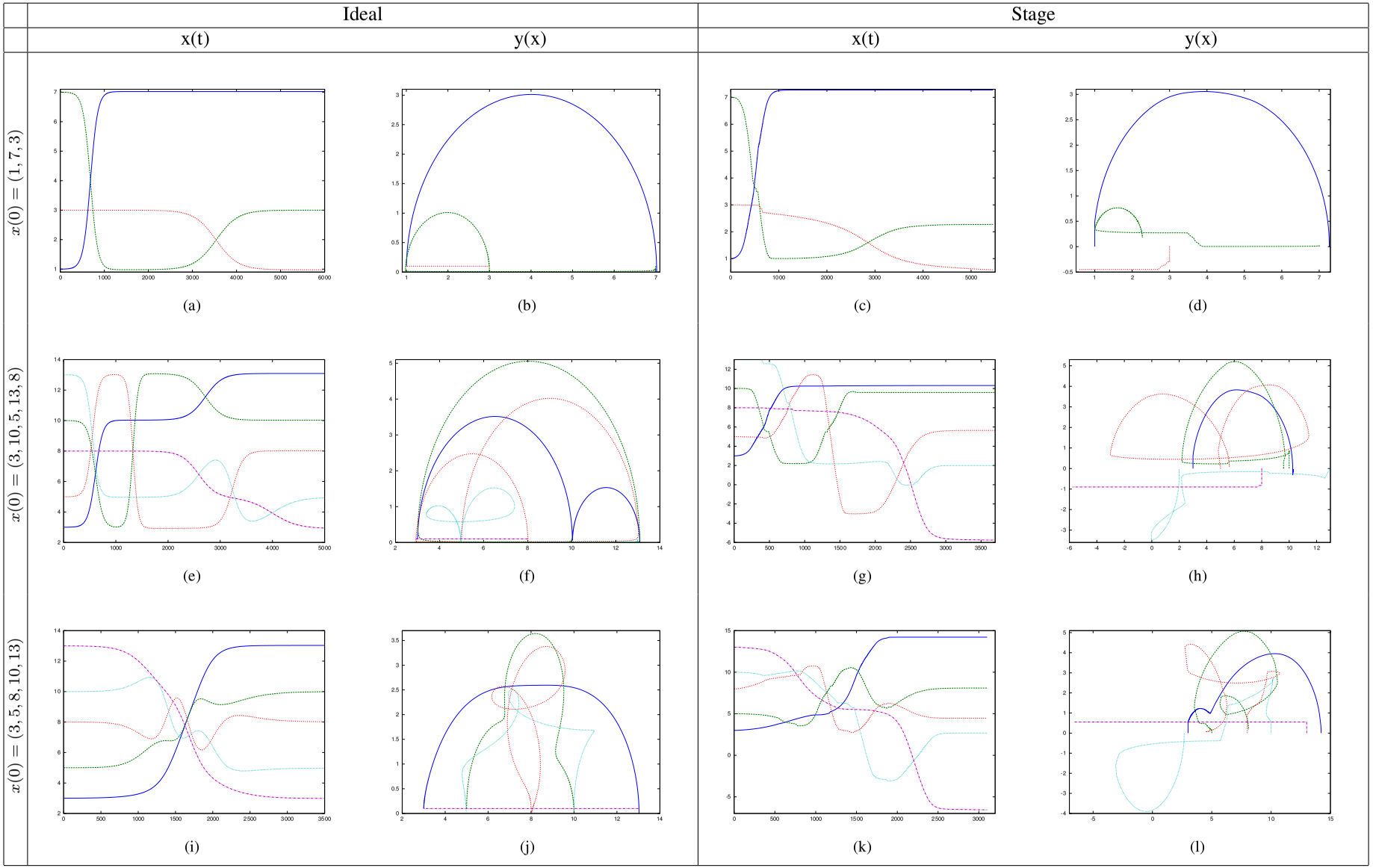


Figure 4. Robot trajectories for different initial conditions. Every row of figures has robots positioned at specified x-coordinates with $y = 0.01$ for all robots. $x(t)$ graphs plot x coordinate(vertical axis) against time (horizontal axis). $y(x)$ graphs plot y coordinate (vertical axis) against x-coordinate (horizontal axis). Disks denote ends of trajectories on $y(x)$ graphs. Note the difference in scales on the graphs.