Group Control and Kernels: The 1-D Equigrouping Problem

Daniel Yamins, Stephen Waydo, and Navin Khaneja

Abstract— **One of the canonical problems of group control is to find local rules through which agents construct specified configurations from arbitrary initial positions. In this paper, we introduce and provide several solutions to the 1-d equigrouping problem, a simple but instructive version of the general spatial configuration problem. We show how deterministic solutions are possible on the linear lattice but not the circle, while the reverse is the case for simple probabilistic solutions. We determine a lower bound on the amount and type of information required by any solution, and relate this information to the geometry of the underlying lattice. Finally, we introduce a concept of an interaction *kernel*, a tool for investigating algorithms in depth. We use the kernel theory to derive several general facts that characterize the group behavior of all deterministic equigrouping solutions, providing a theoretical framework for algorithm design and analysis that may generalize to more complex group control problems.**

I. Introduction

One of the canonical problems of group control of multi-agent systems is to find local algorithms that construct configurations of interest starting from arbitrary initial positions [1], [2], [3], [4], [5], [6], [7], [8]. In this paper, we analyze a very simple version of this problem.

Consider a one-dimensional lattice. Two point-agents placed on this lattice are said to be *in the same group* if all lattice points between the two agents' positions are occupied by other agents. Conversely, two agents are *separated* if there is at least one unoccupied lattice point between them. For each positive integer p, the one-dimensional p-equigrouping problem consists of finding local algorithms which for all m will take any initial configuration of $m \times p$ agents at arbitrary positions into one in which there are m separate groups of p agents each.

1-d equigrouping is a useful problem to study because it illustrates several key difficulties in the local construction of global patterns. A solution has to join every agent up to a p-group even though any given agent cannot locally determine at any given moment whether it should be grouped with the agent on its right or left. On top of this difficulty, if the algorithm groups a given agent with that to its left, then what happens when the algorithm applies the same "directive" to that agent on the left? How does this apparently "recursive" loop end? Furthermore, suppose that a given p-group does succeed in forming. This p-group might be erroneous because it is not made up of agents $pn, pn+1, \ldots, p(n+1)-1$ but is instead a p-group in the wrong frame. Such a group has to be broken down and its constituent agents regrouped. These are inherent and crucial difficulties general to the spatial configuration problems. Finding *ad hoc* an algorithm that solves such a problem – not only the local update rules, but also how much and what kind of information the agents must have available – is often a frustrating exercise in trial-and-error. The equigrouping problem is also interesting because it naturally generalizes to clustering and simple pattern formation in higher dimensions, and on spaces with different underlying geometry (such as the circle or sphere).

In this paper we A) present a simple solution to equigrouping on the line, and analyze the failure of these solutions on the circle, B) provide a conceptual framework for systematic analysis of deterministic solutions $\mathcal{F}_p$, C) calculate lower bounds on the amount and type of information needed by the agents to carry out any solution, and D) present a large class of probabilistic solutions on the circle, and analyze the failure of these algorithms on the line. It will turn out that the technique of partitioning the agents into subgroups which non-trivially interact – so-called "interaction kernels" – provides a natural way to analyze the structure of solutions.

II. Basic Definitions

Following the approach in [9], denote by X an initial configuration of the agents on the line, and let $a_1, \ldots a_n$ be a listing of the agents A in X. Identifying $\mathcal{L}$ with the integers $\mathbb{Z}$, we denote by $pos(a, X)$ the integral value of the lattice point at which agent a is located in configuration X. Hence, $pos(a, X) > pos(b, X)$ indicates that a is to the right of b in X. Given a configuration X, and a set of agents $B = \{a_1, \ldots, a_k\} \subset A$, let $X|_B$ denote the configuration produced by deleting agents not in B. $|X|$ denotes the integer number of agents in X.

For a given agent $a \in X$, let $b_r(a, X) \subset X$ be the ball of radius r around a in X – the r lattice points to the left and r lattice points to the right of a, together with whatever agents are at those points. Let $f(a_i, X)$ be any operator given by

$$f : b_r(a_i, X) \mapsto s,$$

in which s is a lattice segment identical in which agent a can have moved by at most one lattice position. We do not allow two agents to occupy identical positions, and so agents cannot cross positions. In the case that $f(a, X)$ "moves" the agent a to the left, we write $[f(a, X)] = L$; and use analagous notation of R and S to denote right and stationary movement respectively. We require f to be identical for all agents a_i, except the right and left most agents $re(X)$ and $le(X)$, respectively. In fact, we allow $f(\{re(X), le(X)\}, X)$ to be different from $f(a, X)$ where a is not an end-agent, corresponding to the idea of giving agents line-of-sight information about whether or not they

D. Yamins and N. Khaneja are with Division of Engineering and Applied Science, Harvard University, Cambridge, MA 02138, USA `{dyamins,khaneja}@deas.harvard.edu`

S. Waydo is with the Division of Control and Dynamical Systems, California Institute of Technology, Pasadena, CA 91125, USA `waydo@cds.caltech.edu`

have neighbors to their left and right (at whatever distance). Denote the (possibly different) left and right maps by f_l, f_r. We allow f to be probabilistically specified by attaching to each possible configuration of the agent's $b_r(a)$ probabilities p_l, p_r of moving to the left and right, and probability $1-p_l-p_r$ to remaining still.[1] We require f to have a finite well-defined information radius r (the size of the largest ball $b_r(a, X)$ from f can draw information). This is denoted $r(f)$.

We've defined f on a local ball around a given agent; we can "globalize" this action to all of X in an obvious way by taking X to a configuration in which $b_r(a, X)$ has been replaced with s; that is

$$f(a, X) = (X \setminus b_r(a, X)) \oplus s.$$

Let $\mathcal{A}$ be the set of all infinite sequences of agent-labels such that each agent a_i appears infinitely many times. These *allowable semantic strings* correspond exactly to the UNITY semantics described in [10]. We say that f is a solution to the p-equigrouping problem if for all such X with $m \times p$ agents for any m and each $A = (a_1, \ldots, a_n, \ldots) \in \mathcal{A}$, the sequence of compositions

$$\bigcirc_i f(a, \cdot) = f(a_n, (\ldots (f(a_1, X) \ldots)$$

applied to X converges to a fixed equigrouped configuraiton with probability 1. In other words, if we let P_n be the probability that $\bigcirc_i f(a, \cdot)$ is in C_p and remains fixed under any possible further application of f, then $lim_{n \to \infty} P_n = 1$. Let $\mathcal{F}_p$ denote the space of solutions to p-equigrouping. (Henceforth we will use the notation $f_n^s(X)$ to denote the action of the first n steps of f as scheduled by the semantic string s, starting at initial configuration X.)

Given a configuration of agents X, we say that $b \subset agents(X)$ is a j-group if it is a consecutive set of exactly j agents. Denote the right-most agent $re(b)$ and the left-most agent $le(b)$. Intuitively, just as individual agents can (and must) have one of three behaviors (L, R, or S) under any given algorithm f (namely, L, R or S), sets of agents could as well. The formal definition of group behavior is:

Definition 1 Let g be a subset of the set of agents in a given configuration X. Suppose also that g contains no end-agents.

1) For a given algorithm f and semantic string s, we say (f, s) *moves g to the left* if for any positive integer n, there exists m_n, another positive integer, such that the left-end agent after action of (f, s) for m_n steps (that is, $le(f^s_{m_n}(g))$) has been translated to the left of its original configuration (that is, $le(g)$) by more than n places, i.e.

$$pos(le(f^s_{m_n}(g))) - pos(le(g)) < -n;$$

and similarly for the right-end agents. If this possibility holds then we write $[g, f, s] = L$.

2) The definition of (f, s) *moving g to the right* is the same except the end points get translated to the right, i.e.

$$pos(le(f^s_{m_n}(g))) - pos(le(g)) > n;$$

and similarly for the right-end agents. If this possibility holds then we write $[g, f, s] = R$.

3) The other possibility is that under (f, s) the group doesn't move. Formally, we say that (f, s) *stays g in place* if for all m the end-points of $f_s^{(m)}$ have moved no further than some fixed distance d from initial points. If this possibility holds, we write $[g, f, s] = S$.

Though L, R, S are mutually exclusive and exhaustive behaviors at the level of the individual agent, it is not *a priori* clear whether this remains true for groups of agents. We will see below an important class of situations in which it does.

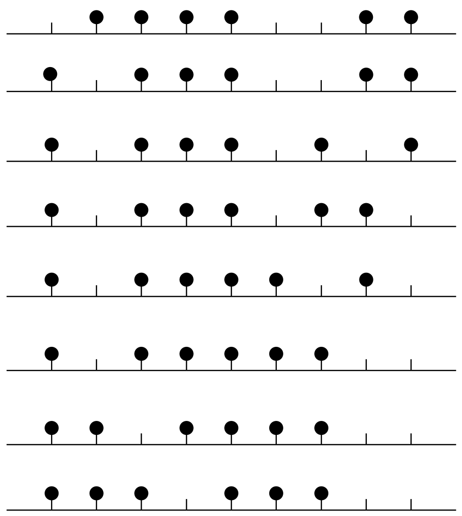
III. A Solution on $\mathcal{L}$

We now present a deterministic algorithm that solves equigrouping problem on the line.

Algorithm 1 For each p, define the algorithm $F(p)$ locally to any given agent a by the rules

1) Suppose a is NOT the first agent to the left along the whole lattice $\mathcal{L}$, but IS the left-most agent of a local group of agents. Then if that local group has size unequal to p, move left, i.e. $[F(p)(a, X)] = L$. If the size of the local group is equal to p, then the action is to stay, i.e. $[F(p)(a, X)] = S$.
2) Suppose a is the first agent to the left along the whole lattice $\mathcal{L}$. If a is in a group of size greater than p, the action is L. If the group's size is $\leq p$, the action is S.
3) In all other cases, agent a's action is S, to stay.

To give the reader a better sense of how this algorithm works, let $p = 3, m = 2$ and consider the initial configuration X in which four consecutive agents are to the left of two consecutive agents. Under the semantic generated by reading off agents repeatedly from left to right, $F(3)$ converges to a solution in 7 steps:



[1]If a given motion is unavailable since the adjacent position in that direction is already occupied, the probability associated with that motion is automatically 0. Deterministic algorithms are simply the special case in which one of $p_l, p_r, 1 - p_r - p_p$ is 1 and the others zero.

Proposition 1 *For each p, $F(p)$ is a solution to p-equigrouping.*

A proof of proposition 1 appears in section V.

Given any algorithm f, we can define a *mirror algorithm*, f_M, by

$$f_M(a_i, X) = \rho(f(a_i, \rho(X)))$$

where ρ is the reflection of X around a_i. If $f \in \mathcal{F}_p$, for some p then $f_M \in \mathcal{F}_p$. This is because any configuration in C_p is still in C_p if it is viewed from "behind" as opposed to from "the front." Hence, the "left" version of Algorithm 1 has a "right version" mirror.

Notice that in Algorithm 1 the information radius $r(F(p)) \geq p$. Also, notice that the behavior of (at least one of) the end-most agents is different from the rest – that is, the left-most agent must *know* that it is an end-agent. It turns out that these facts are always true, as we will see in the next section. Notice also that all the various j-groups travel in the same direction – either they all travel to the left, or to the right (as in the mirror algorithm). It can be shown – by exhibiting several other algorithms – that this is not always true. In other words, that there are algorithms in $\mathcal{F}_p$ which send different-sized groups in different directions. However, there is in fact a unifying relationship between the directions of the various groups, as we shall see in section V.

IV. Basic Theory

First, we present several simple results which provide motivation for the rest of the work.

Proposition 2 *Let f be a solution to p-grouping. Then the information radius of f must be at least p.*

Proof: Suppose that $f \in \mathcal{F}_p$ and $r(f) < p$. Then consider X_1 in which there are p agents in a row, and X_2 in which there are $2p$ agents in a row. On the one hand, f must fix X_1 for any semantic. That is because X_1 is already in the unique (up to quotienting by position) solution. Let a_l and a_r be the left and right-end agents, respectively. We have that

$$[f_l(a_l, X_1)] = [f_r(a_r, X_1)] = S.$$

On the other hand, X_2 is not solved. However,

$$b_{p-1}(a_l, X_1) = b_{p-1}(a_l, X_2)$$

and

$$b_{p-1}(a_r, X_1) = b_{p-1}(a_r, X_2).$$

Hence

$$[f_l(a_l, X_2)] = [f_l(a_l, X_1)] = S = [f_r(a_r, X_1)] = [f_r(a_r, X_2)].$$

But then f (in any semantic) fixes X_2 as well. But therefore f is not a solution, yielding a contradiction. ■

Proposition 3 *There are no deterministic algorithms f such that*

$$f_l = f = f_r$$

as maps of configurations.

Proof: Let $p \in \mathbb{N}$ be arbitrary and suppose the opposition of the assertion: namely that $f_l(a, X) = f(a, X)$ for all pairs (a, X) in which a is the last agent on the left within its $r(f)$-radius, and $f_r(a, X) = f(a, X)$ for all a, X in which a is the last agent on the right within its $r(f)$-radius. Then consider the configuration X^* made up of p agents isolated from each other by more than $r + 1$ spaces on either side, where r is the information diameter of f. Let s be a semantic consisting of repeated iteration of any ordering $a_1, \ldots, a_p$ of the agents, i.e. $s = (a_1, \ldots, a_p, a_1, \ldots, a_p, \ldots,)$. Since f can only move any agent by at most one step, we have that after the first step of f applied to a_1, that agent $d(f(a_1), a_i) > r$ is still true for all i, i.e. a_1 is still isolated, as are all the other agents. But then

$$[f(a_1, X)] = [f(a_2, f(a_1, X))].$$

Therefore after all p agents "go", the result is a translate of the original configuration by $[f(a_i, X)]$. Thus f_s does not converge to an equigrouped solution on X^*, which in this case would have been one p-group, a contradiction to the assumption that f is a solution. ■

We've proved here that the right or left end agent (or both) must in the limited case of being totally isolated, know that it is an end-agent. In other words, some kind of extra information – requiring the ability to communicate across the infinitely long line – is required.

Propositions 2 and 3 are simple examples of more general statements that can be made about deterministic solutions to equigrouping.

V. Interaction Kernels

There is a limit to the power of the results one can derive using the naive techniques of the previous section. To delve deeper into the structure of solutions, it would be useful to decompose algorithms into some type of irreducible units whose behaviors could be analyzed separately and then put together through interactions to form a more comprehensive theory. To this end, we introduce the *interaction kernel.*

Definition 2 Let g be a set of agents on the line $\mathcal{L}$ and f be a deterministic algorithm that solves equigrouping. Then g is a *prekernel* for a given semantic string s if there is an integer $l \in \mathbb{N}$ such that when isolated, given any $m \in \mathbb{N}$, there is an integer $n \geq m$ such that at stage n, consecutive elements of $f^s_n(g)$ are within distance l of each other. In addition, a set of agents g is *irreducible with respect to* (f, s) if for no consecutive decomposition $g = g_1 \oplus g_2$ is $f^s_n(X) = f^{s|g_1}_n(g_1) \oplus f^{s|g_2}_n(g_2)$. A *kernel* is an irreducible prekernel.

In other words, an interaction kernel is a group which, when isolated, "stays together" and which cannot be written as the direct sum of two subgroups which also stay together. For our purposes, the important fact about kernels is that, with a suitably restricted class of semantic strings, they have well-defined, mutually exhaustive and exclusive group-level behaviors. That is:

Proposition 4 ***[Kernel Behavior]*** *Suppose that the semantic string s is composed of iteration of the finite semantic string $\hat{s}$, i.e. $s = (\hat{s}, \hat{s}, \hat{s} \ldots)$ and that g is a kernel for (f, s). Then there is a unique behavior $B \in \{L, R, S\}$ such that $[g, f, s] = B$.*

Proof: For a given (deterministic) algorithm f, consider the set of possible configurations of isolated j-groups within distance l, up to translation. Call this set $R_{j,l}$. It is a finite set, as can easily be seen by counting. Given a configuration X in absolute positions, let $\tilde{X}$ denote the equivalence class of X in $R_{j,l}$.

Because g is a kernel for (f, s), the repeated iterative application of f^s to g never "breaks up the group" forever, so to speak. Denote $X_t = f^s_t(g)$, and let $t_1, \ldots, t_n, \ldots$ be the times when agents of g are in a j-group configuration. Then, with respect to relative position, the map p defined by $p(X_{t_i}) = X_{t_{i+1}}$ induces a map

$$p^* : R_{j,l} \times \mathbb{Z}_k \to R_{j,l} \times \mathbb{Z}_k$$

where k is the length of $\hat{s}$, in which the $R_{j,l}$ component is the relative configuration of X_t up to translation, and the $\mathbb{Z}_k$ component is the step along $\hat{s}$ that the algorithm is at, that is, the residue of $t_i \ mod \ k$.

Since $R_{j,l}$ and $\mathbb{Z}_k$ are finite, iteration of p^* must hit at least one configuration more than once. In particular, let (r, i) be the first configuration such that p^* hits it twice. Suppose that (r, i) is first hit at time τ_1 and then again at time τ_2. The crucial point is that hereafter the update becomes periodic, that is,

$$\tilde{X}_{\tau_1+k}(g) = \tilde{X}_{\tau_2+k}(g)$$

and in particular

$$\tilde{X}_{\tau_1+m(\tau_2-\tau_1)}(g) = r$$

for all $m \geq 0$.

Now, compare X_{τ_1} with X_{τ_2}. Since they both have the same relative positions, there are three possibilities: X_{τ_2} is either a left-translate of X_{τ_1}, a right-translate of it, or is identical to it. Hence we can capture the relative position of X_{τ_2} with respect to X_{τ_1} in a single integer – namely

$$n = pos(a, X_{\tau_2}) - pos(a, X_{\tau_1})$$

which is the same for every agent $a \in g$. Since the update is time-independent, we have that

$$n = pos(a, X_{\tau_1+(i+1)(\tau_2-\tau_1)}) - pos(a, X_{\tau+i(\tau_2-\tau_1)})$$

for all i (and of course all $a \in g$). Hence we can identify a well-defined integer n with (g, f, s), and define the speed

$$n(g, f, s) = \frac{n}{\tau_2 - \tau_1}.$$

Finally, note that if $n(g, f, s) > 0$, then $[g, f, s] = R$ and uniquely so; if $n(g, f, s) < 0$, then $[g, f, s] = L$, also uniquely; and if $n(g, f, s) = 0$, then $[g, f, s] = S$, similarly unique. ■

One thing to note is that, using the same proof technique, we can show that though a prekernel is required only repeatedly to return to within a bounded region, this in fact implies that the agents are *always* within some bounded distance of each other.

Proposition 4 means the kernel behaviors are well defined, and in fact possess well-defined speeds of motion. For notational clarity later on, we define the *periodic set* $per(g, f, s)$ for an (f, s)-kernel g to be the set of configurations that g hits after τ_1, that is, once the kernel becomes periodic. In addition, let $\mathcal{S}^j$ be the set of all pairs (g, s) for which g is a middle group of size j that is an (f, s)-kernel, and s is iteratively generated. Let $\mathcal{S}^j_l$ and $\mathcal{S}^j_r$ be analagously defined for left- and right-groups respectively.

Suppose g and g' are two (f, s)-kernels and are periodic configurations, i.e. $g \in per(g, f, s)$ and $g' \in per(g', f, s)$. We can form various sums $g \oplus g'$ in which the agents of g are placed to the right, or to the left, of the agents in g', at various distances. If g is placed to the left of g' and $n(g, f, s) \leq n(g', f, s)$ – and treated as isolated agents (but not end-groups) – then g will never interact with g'. On the other hand, if $n(g, f, s) > n(g', f, s)$ then the two groups will come into contact – that is, there will be a time such that $le(g')$ and $re(g)$ will be within distance $r(f)$ of each other, and after which $f^s_t(g \oplus g') \neq f^s_t(g) \oplus f^s_t(g')$. The resulting conglomerate might become one larger kernel, break down into several others, or perhaps somehow not generate any kernels at all. Similar, but more complicated scenarios are imaginable with more than two starting kernels.

Whenever kernels exist, we can consider the idea of tracking the "interaction pattern" of kernel formations and break-ups. It can be shown in fact, that in a certain sense, kernels always exist – and that algorithm evolution is always the interaction of kernels.

Proposition 5 ***[Kernel Existence]*** *For size-j configuration g, and iteratively generated semantic s, there is a time t and a consecutive decomposition*

$$g = g^t_1 \oplus \ldots \oplus g^t_m$$

such that each $f^s_t(g)|_{g^t_i}$ is a periodic configuration of an $(f, s|_{g^t_i})$-kernel, and such that under the evolution of f each kernel survives for at least one kernel-period.

In other words, a well-defined kernel-pattern repeatedly arises from *any* configurtion; these kernels interact, agglomerate, and break down. There are much stronger results than this, which we will develop in future work.

Proof: Let the semantic s be generated by $\hat{s}$. Consider the agents of g: they are $\{a_1, \ldots, a_j\}$ from left to right. Now, let $1 \leq i_1, \ldots, i_k \leq j - 1$ be a maximal set of increasing integers such that in the evolution of g under (f, s), given any $m \in \mathbb{N}$, there is a time t_m such that for each j, the agent a_{i_j} at time t_m is at least distance m from its right neighbor, that is,

$$d(f^s_{t_m}|_{a_{i_j}}, f^s_{t_m}|_{a_{i_j}+1}) > m.$$

In other words, the groups $g'_1 = (a_1, \ldots, a_{i_1})$, $g'_2 = (a_{i_1+1}, \ldots a_{i_2}), \ldots, g'_n = (a_{i_k+1}, \ldots, a_j)$ get arbitrarily far away from each other, but then all the agents within the groups themselves remain within a fixed distance, say δ_i, whenever the inter-group distance is larger than some number, say η. Let δ_i be this fixed distance for each group g'_i. Then define $m_1 = max\{2|R_{|g'_i|,\delta_i}| \times |\hat{s}|\}$ and let $m = m_1 + \eta$. After t_m steps, under the evolution of (f, s), the g'_i will be separated by more than m steps. Over the next m steps, the intragroup distance of the g'_i *remains* less than δ_i since the inter-group distance will necessarily be larger than η. But then, via arguments identical to those used in proving proposition 4, periodic configurations get hit, and we see that each of the g'_i must be a sum of non-interacting kernels, each of which "has time" to execute at least one kernel period. This is what we wished to prove. ■

This theory is useful in understanding Algorithm 1. First notice that $F(p)$, under any semantic string s, reduces to $F(p)$ under strings which only activate left-most agents of groups. That is because agents which are not left-most agents of any given group cannot move at all. In this particular subset of semantics, the kernel structure and interaction pattern of Algorithm 1 is easy to calculate.[2]

Suppose a group g does not contain the left-most agent $le(X)$. If g contains fewer than p agents, it is obviously a sum of one-kernels. If it contains p agents, it is a single p-kernel. For g containing $p+1$ or more agents, g decomposes as $g' \oplus P$ where P is a p-kernel composed of the right-most p agents in g. Any 1-kernel will eventually move to the left, as it will be eventually a left-most agent, whereas any agent in a p-kernel will remain fixed. Thus, a 1-kernel interacting with a p-kernel from the right will always produce the pattern $(p, 1) \rightarrow (1, p)$ of kernel sizes.

Now suppose g does contain the left-most agent $le(X)$. If g contains p or fewer agents, it is a sum of 1-kernels. If g contains more than p agents, it clearly decomposes into kernels $P^{+1} \oplus g'$ where P^{+1} is a $p+1$-kernel composed of the left-most agents of g. A kernel interacting with g from the right will always produce another such g. In any case, g will always break down into the kernel decomposition $le(X) \oplus (g \setminus le(X))$ (with at least one space between them) and since $g \setminus le(X)$ does not contain the left-most agent, the previous analysis applies to it. Now $g \setminus le(X)$ contains at least p agents, so therefore a p-kernel exists at its right end.

This calculation allows us to provide a short and conceptually satisfying proof to proposition 1.

Proof: (Proposition 1) We proceed by induction on m, where mp is the number of agents. When $m = 1$, the algorithm clearly works, since $le(X)$ will always be fixed, and agents to the right will always move until they are fixed by being consecutive to $le(X)$. For $m > 1$, if a p-kernel arises at some time, then because 1-kernels always eventually move left and because of the $(p, 1) \rightarrow (1, p)$ interaction, we see that a p-kernel P must eventually get fixed at the right-most end. We can then apply this same reasoning again to $X \setminus P$, and finish by induction. It only remains to show that a p-kernel always arises when $m > 1$. Suppose otherwise. If the left-end P^{+1} kernel situation arose, then a p-kernel arises shortly thereafter (as we saw above), so this is ruled out. But then all agents are in 1-kernels at all times. Then $le(X)$ would have to remain fixed, where as all other agents would have to move indefinately, a contradiction. ■

[2]This semantic situation itself would be difficult to establish in practice since different initial conditions would require different semantic patterns. We will see that this won't affect our results.

We proved the result for restricted semantics activating only left-end agents; but because $F(p)$ reduces to this case, the result applies in the case of the most general UNITY semantics.

Propositions 4 and 5 apply for any algorithm (with iteratively generated semantics), and are not specific to solutions of equigrouping. The important question then is: given that kernels exist, and have well-defined group behaviors, what can we learn about the kernel behavior of solutions to equigrouping? Does the fact of being a solution impose any specific structure on the behavior of kernels? The answer is yes – ultimately, this is the whole point of introducing kernels in the first place. The following theorem collects some of the simplest results along these lines.

Theorem 1 *Let $f \in \mathcal{F}_p$ be deterministic. Then:*

1) *For no iteratively generated semantic strings s_l, s_r, s_m can j-groups g_l, g_r, g_m be kernels of $(f_l, s_l), (f_r, s_r), (f, s_m)$ respectively such that*
$$n(g_l) \leq n(g_m) \leq n(g_r).$$
2) *Suppose that $1 \leq j \leq p-1$, g^j is a j-group kernel for (f_r, s_r), and g^{p-j} is a $p-j$-group kernel for (f_l, s_l). Then*
$$n(g^j, f_r, s_r) < n(g^{p-j}, f_l, s_l).$$
3) *If there is $(g, s) \in \mathcal{S}^j$ with $n(g, f, s) = 0$, then either*
   - *For all $(g', s) \in \mathcal{S}^j_l$, $n(g', f, s) > 0$ or*
   - *For all $(g', s) \in \mathcal{S}^j_r$, $n(g', s) < 0$.*

   This situation is called "end-state squeezing."
4) *Suppose $(g, s_{1,2}) \in \mathcal{S}^j$ and $per(g, f, s_1) \cap per(g, f, s_2) \neq 0$. Then either*
$$sign[n(g, f, s_1)] = sign[n(g, f, s_2)]$$
   or end-state squeezing occurs.

The proof of the theorem is given in the appendix. Proposition 3 is a special case of theorem 1.1, that is, that end-state information is required for a more general setting, and therefore more robustly, than previously shown. Part two provides a relationship between the (end-state) behavior of a j-kernel and its complementary piece, a $p - j$ kernel. Part three establishes that one of the behaviors, namely S, when it appears at all, forces a relatively pathological structure on to the algorithm. Finally, the fourth part shows that, as a result of part 3, the apparent dependence of kernel behavior on the choice of semantic string is highly constrained. One of the future steps in this work is to establish sharper and more restrictive necessary conditions on the kernel-interaction pattern as whole for any solution to equigrouping.

VI. Probability and the Circle

We have found a solution and analyzed aspects of the solution space for deterministic algorithms on the line. But what about finding probabilistic solutions? And what about solutions on the circle? It turns out that there is an interesting link between these two questions.

There are natural maps of algorithms from the line $\mathcal{L}$ to the circles C_n. Let $\phi_n : \mathcal{L} \to C_n$ be a wrapping map taking $0 \in \mathcal{L}$ to $0 \in C_n$. That is, if we identify the positions on $\mathcal{L}$ with the integers, then we can identify C_n with $\mathbb{Z}_n$ and ϕ_n identifies with the standard surjection $\mathbb{Z} \to \mathbb{Z}_n$ (the "modulo-n" map). For any agent a in C_n consider (a, Y) where Y is a configuration on the underlying space C_n. If $r < n$, we can lift (a, Y) naturally to a configuration (a, X) over $\mathcal{L}$ by simply lifting the agent a to the smallest reprensentative a' in $\phi^{-1}(pos(a))$ and defining X to look like $Y|_{B_r(a)}$ around a' and be blank elsewhere. Because r is less than n, the agent cannot determine that it is on a circle as opposed to a line. Our goal is to begin to explore solutions f that work for all circles C_n where n is larger than the information radius of f.

Let $\mathcal{AL}$ be space of locally determined algorithms on the line and $\mathcal{AC}_n$ the space of algorithms defined on the n-circle. We then have a map $\Phi_n : \mathcal{AL} \to \mathcal{AC}_n$ given by

$$\Phi_n[f](a, Y) = \phi_n(f(a, X))|_{B_r(a)} \oplus id.$$

Clearly Φ_n is surjective for each n. Considering the set $\Phi^{-1}(f)$ for some f, we get a number of different algorithms on the line, only different in end-state behavior. The "canonical" representative $\hat{f} \in \Phi^{-1}(f)$ is the one that has end-state maps being the same as the non-end-state maps.

Just as we could define $\mathcal{F}_p$ to be the set of solutions to p-equigrouping on the line, let $\mathcal{F}_p^c$ be the set of solutions to p-equigrouping on the circle. We say $f \in \mathcal{F}_p^c$ if for each n and for all p-equigroupable configurations X on $\mathcal{C}_n$, f solves X. Then

Proposition 6 *There are no deterministic $f \in \mathcal{F}_p^c$.*

Proof: Suppose that f is a deterministic solution. Then it should work for all configurations on C_{n^*} where $n^* = p(r + 1)$. Consider the configuration which equally spaces out p agents on C_{n^*} isolated from each other by $r+1$ places. (This underlying space is just big enough to do this.) Then let S be a 1-to-1 ordering of the agents $(a_1, \ldots, a_p)$ and let s be the semantic generated by repeated iteration of this ordering. Just as in the proof of proposition 3, application of f to any agent does not change the fact that all agents are isolated from each other. Hence just as above,

$$[f(a_1, X)] = [f(a_2, f(a_1, X))].$$

This happens for all i, and as above there is a contradiction. Hence any solution must be non-trivially probabilistic. ■
Because no deterministic solutions on the line work, all deterministic solutions are killed by the natural map.

Another way to see proposition 6 is to notice that any deterministic solution f on C_{n^*} lifts under Φ_{n^*} to a deterministic algorithm $\hat{f}$ on $\mathcal{L}$ which would have $\hat{f}_r = \hat{f}_l = \hat{f}$. But any such $\hat{f}$ must just be a translation, and so f just translates X on the circle, and hence is not a solution. In words: the extra information of "being at the end" which is necessary to allow a deterministic algorithm to solve equigrouping on the line is simply unavailable on the circle due to the symmetry, so *no* deterministic algorithm works. Notice that the idea of non-trivial probability should break the symmetry, intuitively.

Algorithm 2 Let f be such that for any given situation (a, X) the probability of motion available to a in X is non-zero except when a is not the end-agent of a p-cluster, and $[f(a, X)] = S$ otherwise (that is, whenever no motion is available or a is the end-agent of a p-cluster). Call this set of algorithms (when considered as algorithms on the circle) $\mathcal{B}_p^c$

Proposition 7 *The algorithms in $\mathcal{B}_p^c$ are solutions to p-equigrouping on the circle for iteratively generated semantics, that is,*

$$\mathcal{B}_p^c \subset \mathcal{F}_p^c$$

when considered for iteratively generated semantics only.

Proof: Let X be any initial condition on the circle $\mathcal{C}_n$ which can be p-equigrouped (that is, it contains a multiple of p number of agents and the circle is big enough). Suppose that there is non-zero probability that $f \in \mathcal{B}_p^c$ does not solve X; then with non-zero probability there is configuration Y which appears infinitely many times (given that the configuration-space of the circle is finite). In fact, because the semantic string is iteratively generated (say by $\hat{s} = (\hat{s}_1, \ldots, \hat{s}_m)$, there is non-zero probability that Y appears at the same time-step (with respect to the semantic generator), say at step $\hat{s}_i \in \hat{s}$. Now, consider the semantic string after some instance of Y at $\hat{s}_i$. It is generated by $(\hat{s}_{i+1}, \ldots, \hat{s}_m, \hat{s}_1, \ldots, \hat{s}_i)$. We can choose l large enough that some string of actions $b_1, \ldots, b_l$ (where each b_j is either right, straight, or left) in which b_j is the action taken by the agent $\hat{s}_{(i+1+j)\ mod\ m}$, solves Y. Now the probability that the actual actions taken by f for the next l steps after any given instance of Y at step $\hat{s}_i$ are equal to the string $b_1, \ldots, b_l$ is some *non-zero* constant. But the string $(\hat{s}_{(i+1+j)\ mod\ m} | 1 \leq j \leq l)$ appears infinitely many times after instances of Y at $\hat{s}_i$. Hence the probability that the actions taken by f after such an instance eventually hit $b_1, \ldots, b_l$ tends to 1. But once $b_1, \ldots, b_l$ happen, the system halts at a equigrouped solution, and hence Y cannot appear again thereafter. Hence it is a contradiction to assume the existence of Y with non-zero probability; i.e. f is a solution. ■

This result confirms the intuition that probabilistic specification somehow breaks the symmetry which dooms the deterministic algorithms to failure on the circle. It turns out that the algorithms of $\mathcal{B}$ can be shown to work for much larger class of semantic strings. We will not prove this result here due to lack of space.

These ideas on the circle beg the question: What are the probabilistic solutions to the problem on the line? Do the solutions above "lift" to the line? The answer in general is no. In particular:

Proposition 8 *When $p > 3$, $\Phi^{-1}(\mathcal{B}_p) \cap \mathcal{F}_p = \emptyset$. That is, the $\mathcal{B}_{p>2}$ do not lift to the line as solutions.*

Proof: Let $\mathcal{X}$ be the set of configurations on the line $\mathcal{L}$ with $m = 1$, i.e. with p agents. Fix a semantic string iteratively generated by $(a_1, \dots, a_n)$ where the a_i list from left to right the agents in X. Now consider the map

$$G : \mathcal{X} \longrightarrow \mathbb{Z}^{p-1}$$

by

$$G(X) = (pos(a_2) - pos(a_1), \dots, pos(a_n) - pos(a_{n-1})).$$

Under the evolution of any $f \in \Phi^{-1}(\mathcal{B}_p)$, $G(f_t^s(X))$ traces out a random walk in the set $\mathcal{G} = \{z \in \mathbb{Z}^{p-1} | z_i > 0\}$, reflecting at any of the boundaries (i.e. when $z_i = 0$ for some i). For f to be a solution on the $\mathcal{L}$ to all elements in $\mathcal{X}$ would mean that for any initial point in $\mathcal{G}$, the random walk starting at that point would have to hit $\mathbf{1} = (1, \dots, 1) \in \mathbb{Z}^{p-1}$ with probability 1. (Once such a path hits $\mathbf{1}$ it becomes stationary, due to the rules of the algorithm.) But in dimensions 3 or higher, random walks on lattices are not ergodic [11]. Hence, the probability that such a path will hits $\mathbf{1}$ is less than 1 when $p \geq 4$. Hence f will not succeed with probability 1 for the semantic string s, and is therefore not a solution. ■

Are there any probabilistic algorithms on the line? Yes: in fact, it turns out that $\Phi^{-1}(\mathcal{B}_2) \subset \mathcal{F}_2$, (a partial converse of proposition 8) but due to limited space we will not prove this here.

What is the relationship between propositions 3 and 8? The first identifies an obstacle to creating deterministic algorithms without specific end information. This "determinism trap" can be overcome on the circle by using probability. However, proposition 8 shows that the determinism trap is not the only obstacle to removing the requirement of extra infinitary information. Because $\mathcal{L}$, unlike the circles, is non-compact, issues of ergodicity become important. On a finite line, as on the circle, this problem would go away.

VII. Appendix

Proof: (Theorem 1)

1) Suppose on the contrary that there are iteratively generated s_l, s_r, s_m and initial configurations g_l, g_r, g_m of j agents each, which are left-, right-, and middle kernels respectively for (f, s_l), (f, s_r) and (f, s), and whose respective speeds satisfy the given inequality. We will construct an unsolved initial configuration and a semantic under which f cannot solve the given configuration. In particular, consider

$$X = g_l \oplus g_m \oplus g_m \oplus \ldots \oplus g_m \oplus g_r$$

in which which g_m is repeated $\frac{lcm(j,p)}{j} - 2$ times. Clearly the configuration as constructed has a multiple of p agents. Now consider the semantic s iteratively generated by $s_l \circ s_m \circ s_r$. Under the pair (f, s) the g_l group of agents is separated far enough from the rest of the agents to its right so that it behaves as the separate (f, s_l) kernel. Similarly for each copy of the g_m group and the g_r group. But then given that $n_l \leq n_m \leq n_r$, we see that the system evolves so that the g_l group never comes into contact with the left-most copy of the g_m group, whereas the right-most copy of the g_m group never comes into contact with the g_r group. In fact, if any of the inequalities are strict, then the groups get increasingly farther apart. Hence X is never solved by f under the given semantic, a contradiction.

2) Suppose again on the contrary that g^j is a size-j left-kernel for (f, s_l) and g^{p-j} a size-$p - j$ right-kernel for (f, s_r). Then consider the configuration $X = g^j \oplus g^{p-j}$. This configuration has p agents. Consider the evolution of it under $(f, < s_l \circ s_r >)$. If the speed of the right group is greater than that of the left group, the two will come into contact and hence f cannot solve X for this semantic, a contradiction.

3) If there exists s (iteratively generated or otherwise) such that $[f, g, s] = S$, then for any configuration g of j agents, either all s_l (not necessarily iteratively generated) for which $[g, f_l, s_l]$ exists have $[g, f_l, s_l] = R$ or all semantic strings s_r for which $[g, f_r, s_r]$ exists have $[g, f_r, s_r] = L$. Suppose otherwise. Let g_m, s_m be a j-agent configuration such that $[g_m, f, s_m] = S$. Let g_l, g_r be j-group agents which have well-defined behavior under s_l, s_r, but such that $[g_l, f_l, s_l] \neq R$, $[g_r, f_r, s_r] \neq L$. Then using the same construction of X as in the first part yields a structure which will never be solved. Now, we can of course apply this to the case of the j-groups being kernels under the various semantics, which gives the result as stated.

In fact, consider the set $v(j, p)$ of $\mathbb{Z}_p$ residues of multiples of j and suppose that for any configuration x of j agents (not as an end group) there is *some* semantic string (iteractively generated or otherwise) staying x. Then for any $m \in v(j, p)$ and $m_1, m_2 > 0$ such that $m_1 + m_2 = m$, either all semantics s_l send any configuration of m_1 agents considered as a left-end group to the right or all semantics send any configuration of m_2 agents considered as a right-end group to the left. This follows for the same reason as above, but using X constructed as

$$X = g_l^{m_1} \oplus g_m^j \oplus g_m^j \oplus \ldots \oplus g_m^j \oplus g_r^{m_2}$$

in which the j-group g_m^j (considered as a middle group) is repeated k times, where $kj + m_1 + m_2$ is some multiple of p. (This is why we require that $m_1 + m_2$ is a p-residue for some multiple of j.)

4) For iteratively generated semantics s_1, s_2 consider the sets P_1, P_2 of periodic kernel configurations for s_1, s_2. Suppose that $x \in P_1 \cap P_2$ and that s_i is generated by $\hat{s}_i$. Then if $n_1 = n(x, f, s_1)$ has a different sign from $n_2 = n(x, f, s_2)$, and suppose wlog that $n_1 > 0$. Then consider the semantic

$$s = (\hat{s}_1, \hat{s}_1, \dots, \hat{s}_1, \hat{s}_2, \hat{s}_2, \dots, \hat{s}_2)$$

where $\hat{s}_1$ is repeated $b_1 = |\frac{lcm(n_1,n_2)}{n_1}|$ times and $\hat{s}_2$ is repeated $b_2 = |\frac{lcm(n_1,n_2)}{n_2}|$ times. The result is that the speed

$$n(g, f, s) = n_1 b_1 + n_2 b_2 = lcm(n_1, n_2) - lcm(n_1, n_2) = 0.$$

That is, g is an (f, s) kernel with speed zero. Hence the previous part applies to give the result. ■

References

[1] C. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model," *Proc. SIGGRAPH*, 1987.

[2] J. Toner and Y. Tu, "Flocks, Herds, and Schools: A Quantitative Theory of Flocking," *Phys. Rev. E*, Vol. 58 No. 4, 1998.

[3] R. Olfati-Saber and R. M. Murray, "Flocking with Obstacle Avoidance," *Proc. IEEE CDC03*, 2003.

[4] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of Groups of Mobile Autonomous Agents Using Nearst Neighbor Rules," *IEEE Transactions on Automatic Control*, Vol. 48, No. 6, 2003.

[5] H. Tanner, A. Jadbabaie, and G. Pappas, "Stable Flocking of Mobile Agents, Parts I and II," *Proc. IEEE CDC03*, 2003.

[6] P. Ogren, E. Fiorelli, and N. E. Leonard, "Formations with Mission: Stable Coordination of Vehicle Group Maneuvers," *Proc. 15th International Symposium on Mathematical Theory of Networks and Systems*, 2002.

[7] F. Zhang and P.S. Krishnaprasad, "Coordinated Orbit Transfer for Satellite Clusters," *Proc. IEEE CDC02*, 2002.

[8] E. Klavins, "Autnomatic Synthesis of Controllers for Distributed Assembly and Formation Forming," *Proc. IEEE Conference on Robotics and Automation*, 2002.

[9] D. Yamins, "A Framework for Hierarchical Agent Systems," *Proc. ALIFE VIII*, 2003.

[10] E. Klavins, "A Formal Model of Multi-Robot Control and Communication Tasks", *Proc. IEEE CDC03*, 2003.

[11] E. W. Montroll, "Random Walks in Multidimensional Spaces, Especially on Periodic Lattices," *J. SIAM*, Vol. 4, 1956.