

Bess L. Walker
(advised by Keith Garfield of the University of Central Florida)
CS 497: Honors Research (overseen by Dr. Antony Hosking)
Semester Report

Necessary background can be found in my technical report covering my summer research activities; a copy is attached.

The primary goal for this semester's research was to modify the leadership behavior I had implemented over the summer in order to increase MAV performance in dead-end environments. The original MAVs did not distinguish between sensory and avoidance radii; sensing a neighbor forced a MAV to take the appropriate action. In most cases, this action was avoidance; the UCF dominance scheme and my own original leadership code got around this by incorporating the avoidance code into our social schemes. By separating these radii, I hypothesized, MAVs could keep strict avoidance behavior for neighbors within the avoidance radius but have social interactions with sensed neighbors outside the avoidance radius. This would allow more genuine behavior separation and more compact code, and improve several aspects of the leadership behavior which I thought were hampered by these radii being identical.

The original leadership behavior was relatively simple. A MAV's leadership value was adjusted according to its success in movement. (MAVs have no memory, so cannot be rewarded only for "positive" movement. The only thing they really know is if they are stuck or not, and that Stuck is Bad.) If a MAV A sensed another MAV B, the mixed leadership/avoidance strategy came into play: if B had the highest leadership of all the MAVs A sensed (including A itself), A was encouraged to move toward B. Otherwise, A avoided B: the directions toward B were taboo for movement.

The most serious problem with this strategy was that it allowed follower-MAVs to collide with each other, since their normal avoidance behavior was overridden to allow them to follow. Additionally, a MAV with high leadership could lead only a very few other MAVs, because each MAV had to be quite close. This made the leadership strategy less effective than it could have been. Separating the sensory and avoidance radii would allow me to deal with avoidance and leadership separately, fixing the collision problem, and would allow the MAVs a wider sensory range, hopefully allowing leaders to gather more followers.

Implementing the separation was a breeze. The MAV simulation has a `run.conf` file from which it takes many important parameters, and I added an avoidance radius parameter. A few tweaks to the code in the MAV sensors and behaviors, and the radii were separate. I could assign behaviors designed to avoid and social behaviors without incorporating the social behavior as just a hack around avoidance.

The specific effects of the separation were exactly as I had hoped. MAVs now followed leaders sensed beyond the avoidance radius, and avoided everything, with no exceptions, within it, solving the follower-collision problem. Adding a little more to the GUI to display leader-follower ties confirmed that leaders were influencing more MAVs.

Unfortunately, the change also destroyed the positive effects of the original leadership implementation. Leadership no longer acted as a scattering mechanism in crowded situations, and, most damningly, no longer enabled MAVs to escape dead ends. After watching not a few “before” and “after” simulations, I came to the conclusion that the loss of these effects was due to the loss of something I called the Pushmi-Pullyu Effect, which I had noted in my summer research but had thought was simply a curiosity. In fact, it turns out, it is what made everything work.

The essence of the Pushmi-Pullyu Effect is that the leader and follower MAV are equally important when escaping a dead end. The leader cannot go into the dead end because it would collide with the follower. But a MAV moves if there is any direction in which it can move, so it is forced in a direction leading out of the dead-end. This movement increases its leadership score. With the original leadership implementation, the follower *can* move towards the leader, and (depending on whether it can sense other goals) often does. Thus it is again in the position to prevent the leader from going back into the dead end, and this dynamic eventually allows the pair to escape. The follower’s “push” is just as important as the leader’s “pull” – which is why, with apologies to Hugh Lofting, I’ve called it the Pushmi-Pullyu Effect.

The new leadership implementation doesn’t have this dynamic, because follower behavior has changed. If the MAVs are within the avoidance radius, they always avoid each other. Thus, although the leader may initially move towards the entrance to the dead end, the follower can’t get inside its avoidance radius to keeping “pushing” it out, because it must strictly obey its own avoidance radius. I am not certain at this time whether or not I can get a Pushmi-Pullyu Effect with the radii separated, or what would have to be changed to achieve it.

A secondary goal for the semester was to run a genetic algorithm on the MAVs’ leadership (and later on, radii) parameters. At this point I had too many parameters to be adjusting them by hand and hoping I’d get good behavior. There were two essential problems with this: first, although I know a bit about the idea of genetic algorithms, I know little about their implementation; second, since it is tied to a GUI, a MAV simulation can take over twenty minutes to run. Before I could contemplate running a genetic algorithm, I had to have a faster way to run the simulations. I had to be able to run the simulation without invoking the GUI.

I had dealt with the slow simulation runtime over the summer because I didn’t need to run very many simulations to come up with data – 20 simulations each in three different environments with leadership on and off. Nevertheless, it took more than a day for the simulations to finish. I had been annoyed then that there wasn’t a non-GUI interface – MASON, the simulation tool used, was designed so that a simulation and its GUI would be separate and the simulation could be run without the GUI, but here I was with a bunch of code that had the two interleaved. The original UCF/IST MAVs had had genetic algorithms applied to determine their parameters, so there must have been a non-GUI version at some point. I assumed that lazy coding over the duration of the project had made the GUI an integral part of the simulation. So I sat down to figure out how to separate them.

So it was with growing confusion that while I saw GUI-related code everywhere, and in fact had modified it over the summer so that MAV leadership levels would be

displayed as color saturation, I couldn't find where it was being called. Finally I took a look in the ant build.xml file and found that, although I was using a command "ant operasimsuite" to run the simulation, and assuming that this ran straight from OperaSimSuite.java, it was in fact running main() in some class called OperaSimSuiteGUI.java. Well, that solved the mystery of why I couldn't find a main() function in OperaSimSuite, and revealed that, as long as I provided a proper main() function, I *could* run the simulation without the GUI. A misleading command name had been fooling me all this time, and now the time to run a simulation was down to around three minutes. Faster would have been nice, but three minutes was much more acceptable than over twenty.

It was at about this point in my research that I realized that the fundamental problem with the new leadership implementation might be exacerbated by poorly-tuned parameters, but it was ultimately caused by the disappearance of the Pushmi-Pullyu Effect. As it was quite late in the semester (I didn't start really working on the research until November, as I was running around trying to find someone at Purdue to make it legitimate), I concentrated on trying to fix that problem, rather than coming up with a genetic algorithm.

I can't say I'm entirely pleased by my work this semester. I started late, and didn't exactly get the results I was hoping for. But I did gain a much increased appreciation of the Pushmi-Pullyu Effect, and after all, research is about discovery. That discovery isn't always what we expect.