Local Routing in a new Indefinitely Scalable Architecture

Trent Small

The University of New Mexico
Albuquerque, NM, 87131-0001
tsmall1@unm.edu

Abstract

Local routing is a problem which most of us face on a daily basis as we move around the cities we live in. This study proposes several routing methods based on road signs in a procedurally generated city which does not assume knowledge of global city structure and shows its overall efficiency in a variety of dense city environments. We show that techniques such as *Intersection-Canalization* allow for this method to be feasible for routing information arbitrarily on an architecture with limited resources.

Introduction

New architecture = new problems

Designed for robustness, the *Movable Feast Machine*(Ackley et al., 2013) (or briefly MFM)is a new computer architecture aimed at spatially distributing computation in a way that can scale indefinitely. This system is rather robust to many hardware errors (for instance, by its distributed nature) and software errors (for instance, by its use of Cellular Automata).

Robustness comes at a price however, as this architecture is drastically different from the tried and true Von Neumann architecture. This makes traditional computer problem solving much more challenging. In fact, thinking traditionally about computer problem solving in the MFM is hardly feasible, particularly problems which need to access any sort of global state. This brings about many problems with traditional computing procedures.

Routing revisited

The Von Neumann architecture supplies hardware for getting information from one end of the machine to the other. Unfortunately, this is not particularly robust as a single fault in the machine will cause the communication route to be severed. This is where the MFM shines, as a fault in one part of the machine allows the rest of the machine to function. However, routing information from one part of the machine to any other becomes more difficult.

Because it is a Cellular Automata on distributed hardware, The MFM memory on the system corresponds to the location of the tile directly. This forces any data moving across an MFM simulation to move physically across the machine as well. The small Event Windows of the MFM also make it difficult to know much about the surroundings of an atom when it is processing an event. The way that the MFM is distributed also makes it impossible to create an absolute addressing system like the Von Neumann architecture exhibits, making absolute routing impossible.

A solution by example

Because the MFM is built as an Artificial Life simulator(Ackley and Small, 2014), much thought was put into creating a system much like the world that we live in. This allows us to examine problems which the Von Neumann architecture has solved and apply real-world techniques in order to solve them on the MFM. Even the humblest of creatures work on solving this spatial routing problem on a daily basis, so many angles of inspiration exist.

One solution that comes to mind is based on the idea of a city. Cities are dense, which allow information and resources to travel only short distances. Cities also exist upon a heavily structured road system which allow information to travel quickly. Cities are also rather diverse, allowing for many different kinds of information to share the same structure. All of these properties are useful aspects when routing inside a computer. Speed and flexibility are two aspects which are deemed important in routing cars described by the *Vehicle Routing Problem* (VRP) (Cordeau et al., 2002). Two more aspects described in (Cordeau et al., 2002) which are good to focus on are accuracy and simplicity. These four aspects are important to keep in mind when routing data as well.

Being a central part of many lives, lots of research goes into the efficiency of cities. Many delivery algorithms exist, but most are more applicable to the real world with their assumption of global state(Lu and Dessouky, 2004)(Clarke and Wright, 1964)(Wiering et al., 2004) (Lämmer and Helbing, 2008)(Gershenson and Rosenblueth, 2009) This is not a luxury that the MFM provides.

There has also been research on road density using cellular automata (Nagel and Schreckenberg, 1992), and even

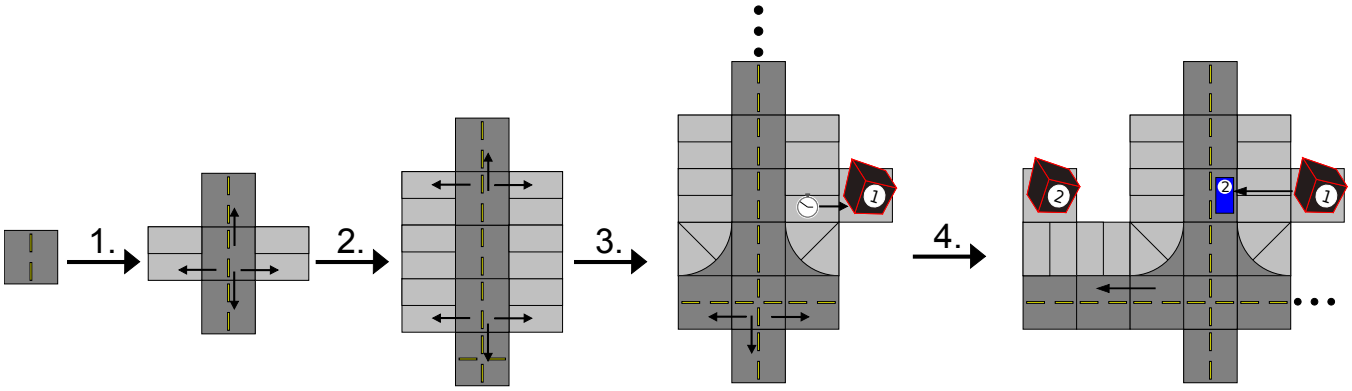

Figure 1: City growth stage: **1.** A street creates streets and sidewalks. **2.** Streets sometimes create intersections. **3.** Intersections create streets. After time, sidewalks create buildings. **4.** The city grows. Buildings create cars which travel along roads to find other buildings.

the effect of controlling traffic lights in a cellular automata city to increase traffic efficiency without collisions (Brockfeld et al., 2001). These simulations are designed to keep cars from colliding, whereas the simulation we are looking to create simply needs to route information as quickly losslessly as possible.

It is also worth it to note that city planners appointed by United Nations Human Settlements Programme are now starting to use tile-based software to plan cities(Parker, 2014).

Design considerations

Unfortunately, there are many properties of real cities which allow for routing to be efficient that we can not take advantage of. Most things that are being routed in the real world are controlled by people whom are able to remember how to get around the city. The MFM does not allow for this luxury since each event can only access 41 sites, each with 96 bits of information. This is nowhere near enough information to remember where building are around the city. Therefore, we must treat every piece of routed information as if they are new to the city and unable to navigate without its help. This brings us to our first design consideration:

Local mapping: There must exist some method which aides information in approaching its destination.

According to (Cordeau et al., 2002), a good routing system has four traits. Therefore, it is natural to use these traits as a guide in our routing system. Thus, we have another design consideration:

Routing traits: Routing must be fast, accurate, simple, and flexible.

Because this software is running on the MFM, we also need to consider the time and thought put into designing a system which is robust to errors. Our city should be able to exist on such hardware and conform to most of these robustness standards. Thus, we have our final design choice to consider:

Keeping robustness in mind: The city must favor robustness over correctness.

Utilizing techniques common to self-organization (Misteli, 2001) will increase the robustness of the city and allows traffic to continue to flow in an organized manner in many destructive scenarios after a short rebuilding stage.

Limitations

Unfortunately, simulating real city traffic in the MFM is not feasible because there are simply too many things that can go wrong. Some compromises must be made to our model in order to gather meaningful information about city routing.

Single-lane swapping

Each street in this model can be seen as a two-way four-lane road. This is a little tricky however, since each street is a single atom. In order to make up for this, when a car moves it may swap its position with either a street or a car that is in the direction it is moving in. This makes for some interesting dynamics since cars waiting at an intersection can move towards the front of their queues. These simple rules allow a two-way one-lane atomic road to simulate a two-way four-lane road.

This is not how real traffic works, and it does end up needlessly using gas while waiting at an intersection, but there has already been research in these kinds of problems using cellular automata (Nagel et al., 1998) which shows that traffic can flow well even in a two-lane environment as long as it follows some simple rules.

Building types

Because there is no notion of absolute addressing in the MFM, there is no way that a car can hope to find a unique

building in this city. In order to compensate for this fact, the model uses *building types*. There are 24 designated types for which a building may emulate. This differs from the real world in that there are many unique buildings, but most real world buildings can be placed into categories for simplicity's sake. For example, most grocery stores are nearly identical, and the closest one to home is often the one most frequented (Hsu et al., 2010).

Map size

Unfortunately, the MFM only provides each atom with 71 general-purpose bits to use for anything an element enginner pleases. If 24 building types are given, this is only enough to allocate (given that a small number of other bits are allocated for other purposes) 2 bits per building type to anything which should make a local map of the nearby city blocks. Used cleverly, we will see that this limitation does not seem to impede routing much, but an MFM simulator which is able to give more bits per atom would have allowed for some experimentation of these map sizes.

Model

The model consists of a *City Growth* stage (which eventually leads to a local equilibrium), and a *Routing* stage. A user begins the first stage by placing a single Element into an empty MFM grid.

City Growth

The city begins with a single ELEMENT_CITY_STREET *(see Fig.1)*, which simply tries to copy itself in a cardinal direction. If the street encounters an Element other than something which belongs in the street (namely an ELEMENT_CITY_CAR, an ELEMENT_CITY_INTERSECTION, or another Street), it replaces that Element with either a Street or an Intersection depending on its Intersection odds parameter. The roads continue to grow indefinitely.

As the streets grow, they place ELEMENT_CITY_SIDEWALK elements in the sites perpendicular to the direction they are traveling. Once a Sidewalk has been created, it waits for a parameterized number of events before creating an ELEMENT_CITY_BUILDING on the side opposite the Street. The sidewalks are produced solely by the Streets and do not reproduce.

Once an ELEMENT_CITY_BUILDING has been created, it begins to reproduce along the Sidewalk. Each building has a TYPE parameter, symbolizing the type of building it is, which corresponds directly to the *building type* limitation. These buildings only grow up to a maximum size, allowing diversity on every city block. Once a building has been created, it also gains the capacity to create an ELEMENT_CITY_CAR in place of a nearby Street.

The ELEMENT_CITY_CAR simply drives along a Street by swapping in the direction of the Street it was created in place of using the single-lane swapping method described above. A car has a limited amount of events before it runs out of gas and is consumed by the system. If a Car has an event next to a Building of the same type as its destination parameter, it is consumed and the remaining fuel is reported.

Once the cars begin moving around the city, they may hit a dead end or the edge of the universe (in which cases they perform a U-turn), or they will run into an intersection. This is where routing takes place. The cars, upon arriving at an intersection, wait for the intersection to make an intelligent decision regarding which direction they will travel in next. The intersection ultimately moves the car onto one of its bordering streets which allows the car to continue in its new direction.

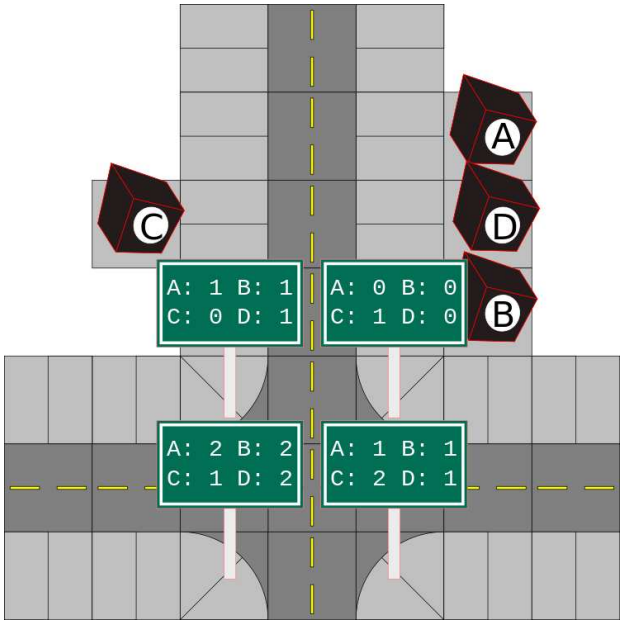

Figure 2: Sidewalks contain a map of their city block distance from every building type.

Routing methods

We will be examining several local routing algorithms in this city.

Random routing

This routing algorithm is the most basic of them all. When an intersection sees a neighboring car, it will choose one of the streets it borders at random which the car did not come from and will send the car in that direction. This routing algorithm is akin to wandering around the city in no organized manner.

Sidewalk-Only routing

This routing algorithm takes advantage of the state inside of the Sidewalks which border the streets. Sidewalks build a

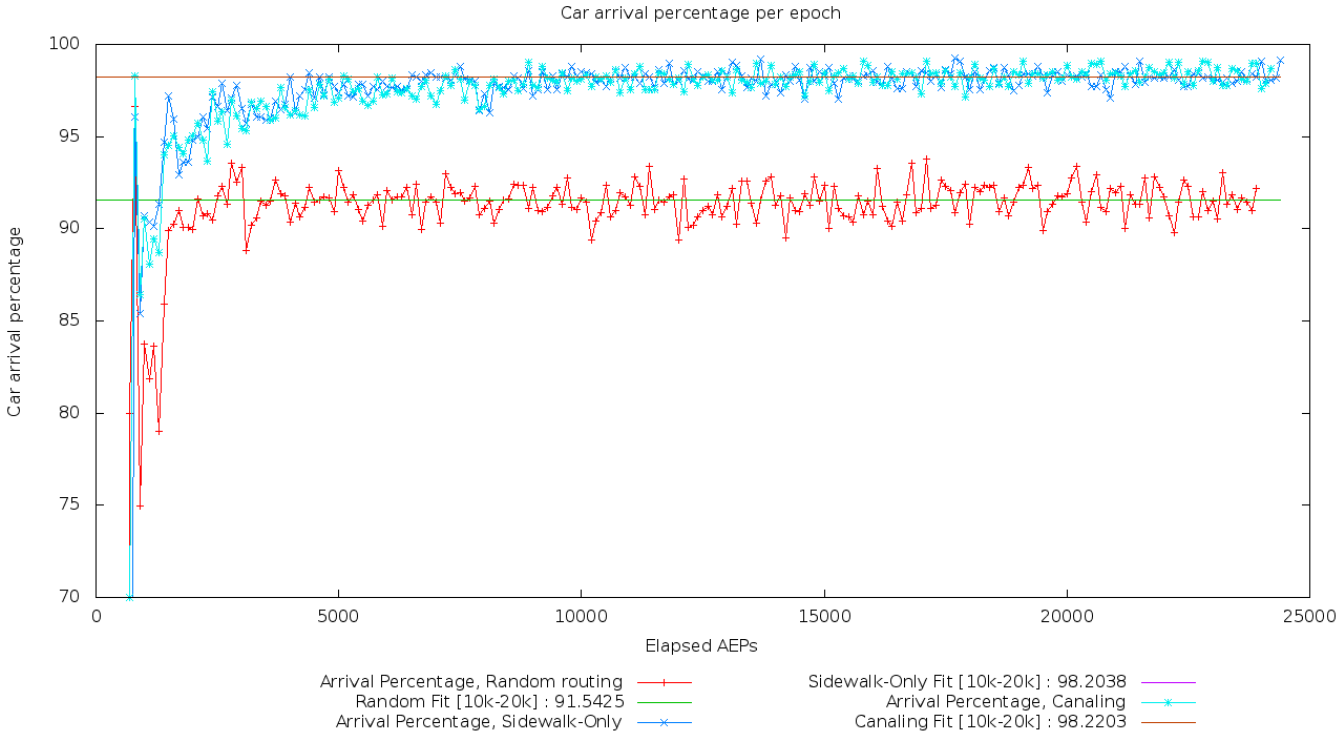

Figure 3: Summary of routing methods in terms of car arrival accuracy, averaged over ten simulated runs.

local map of the buildings near them. They store an n length array of 2-bit numbers, where n is the number of building types in the city. This array stores the distance of this Sidewalk from the building of each type, measured in city blocks (see Fig.2). This local-city-block map is populated using Algorithm 1.

if *I border a building* **then**
  Set my map[building type] to 0
end
for *each building type* t **do**
  Get the minimum m of my neighbor's maps for t
  Set my map[t] to MIN(map[t], m).
  if *There is a Sidewalk across the street from me,* s **then**
    Set my map[t] to MIN(map[t], (s[t]+1)).
  end
end

Algorithm 1: Sidewalk Mapping.

This map is ultimately read by an Intersection when a Car is waiting. The Intersection examines the Sidewalks surrounding neighboring Streets and places the Car on the Street which is between the blocks closest to its destination. If a car just came from this direction, it will pick the second best choice, since making a car perform a U-turn will never help it approach its destination unless it is facing a dead-end.

Unfortunately, sidewalk-Only routing is prone to problems. When an intersection decides to send a car down a dead end, it does not remember this and cars of this type will to be sent to this dead-end every time they arrive.

Intersection-Canalization routing

Intersection-Canalization routing builds upon Sidewalk-Only routing. The intersections are now given a small routing table which remembers the last direction that it sent a car of each type. Canalization prevents the problem that Sidewalk-Only routing creates by re-evalutaing its routing choices less often and trusting its memory of the city. Training traffic lights using methods more complicated than canalization also proves to be successful(Tubaishat et al., 2007) which seems to imply that a smarter traffic light may be more correct.

Results

These three routing algorithms each ran for ten simulated runs, each run generating its own city from a single street atom. The results from each run were then averaged and plotted into Fig.3 and Fig.4. These cities were generated with parameters which placed buildings densely (see Fig.5), as is the case with enviornments like New York City.

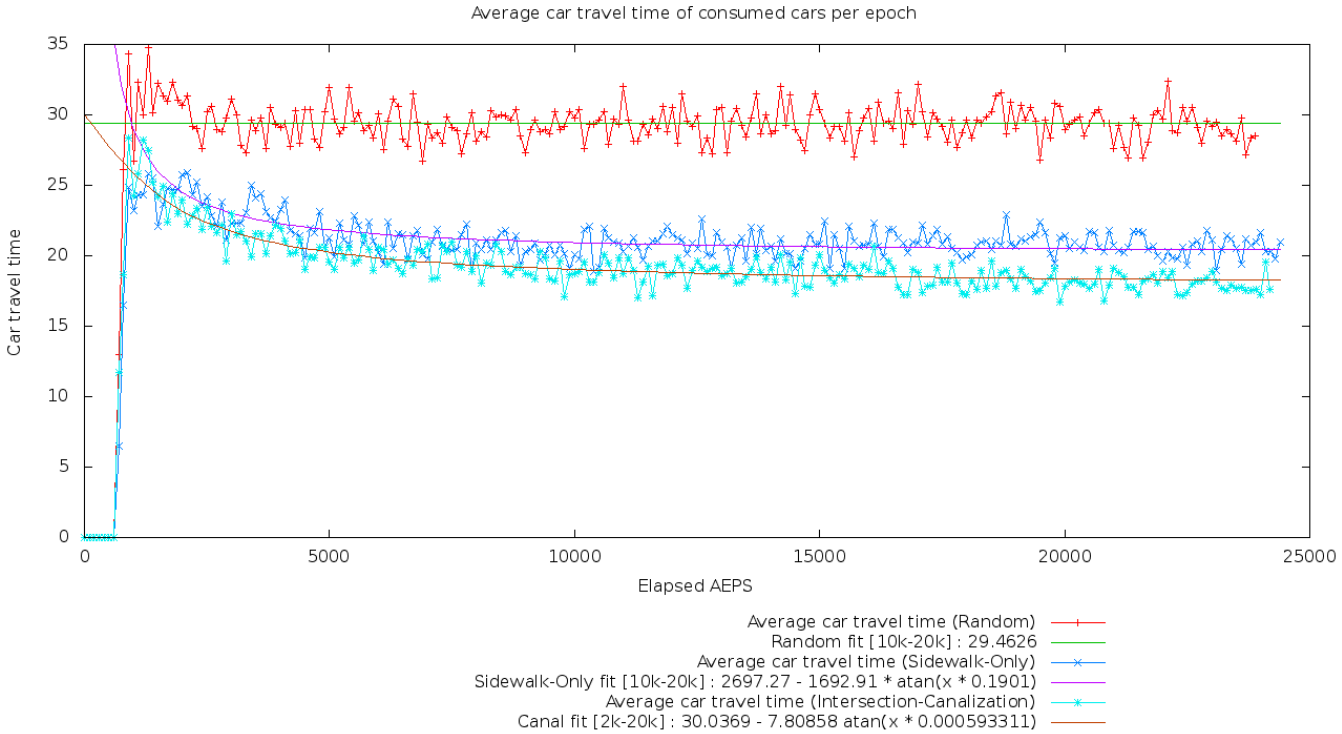

Figure 4: Summary of routing methods in terms of gas usage of correctly routed cars, averaged over ten simulated runs.

Random routing

It may be surprising to hear that randomly routing cars around these cities is an algorithm that performs relatively well. Over ten runs, the limiting behavior of the random routing algorithm suggests that cars reach their destinations 91% of the time without running out of gas. The density of the city is most likely what helps these cars in the long run because, although the building types are uniformly distributed, city blocks are never far apart from blocks which have similar buildings. However, this also means that one in ten dispatched cars does not reach its destination before it runs out of gas.

The cars that do reach their destinations do it in a relatively short amount of time, at an average of 29.4 events after being dispatched.

Sidewalk-Only routing

Routing cars using the Sidewalk-Only technique is an impressive improvement over the Random routing technique. Even with the dead-end faults that this algorithm exhibits described earlier, dispatched cars following road signs reach their destinations at a rate of 98.2% after the city growth stage. This is an improvement from nearly $\frac{1}{10}$ cars not making it to their destinations to less than $\frac{1}{50}$ cars. This algorithm helps nearly five times as many vehicles reach their destination.

Destination accuracy is not the only success shown by Sidewalk-Only routing over Random routing; the average time taken for a car to reach its destination decreases to 20.05 events, which is a 29.11% decrease in gas usage.

Intersection-Canalization routing

Routing cars using the Intersection-Canalization technique yields an improvement over Sidewalk-Only routing, mainly because it avoids sending many cars down dead-ends in the city. However, this technique is only slightly better at Sidewalk-Only routing in getting cars to their destinations, at a rate of 98.22% after the city growth stage and a short training time, an 0.02% increase from Sidewalk-Only routing.

Although cars are routed with only a small increase in correctness, success is found when calculating the average time taken for a car to reach its destination, which in this case is 17.7 events. This is a 15% decrease in gas usage from Sidewalk-Only routing. This means that on average, every car in the city is can operate with only 85% as much fuel as was needed in Sidewalk-Only routing.

Conclusion

These routing algorithms show that, in an environment which is new to the traffic being routed or has very limited

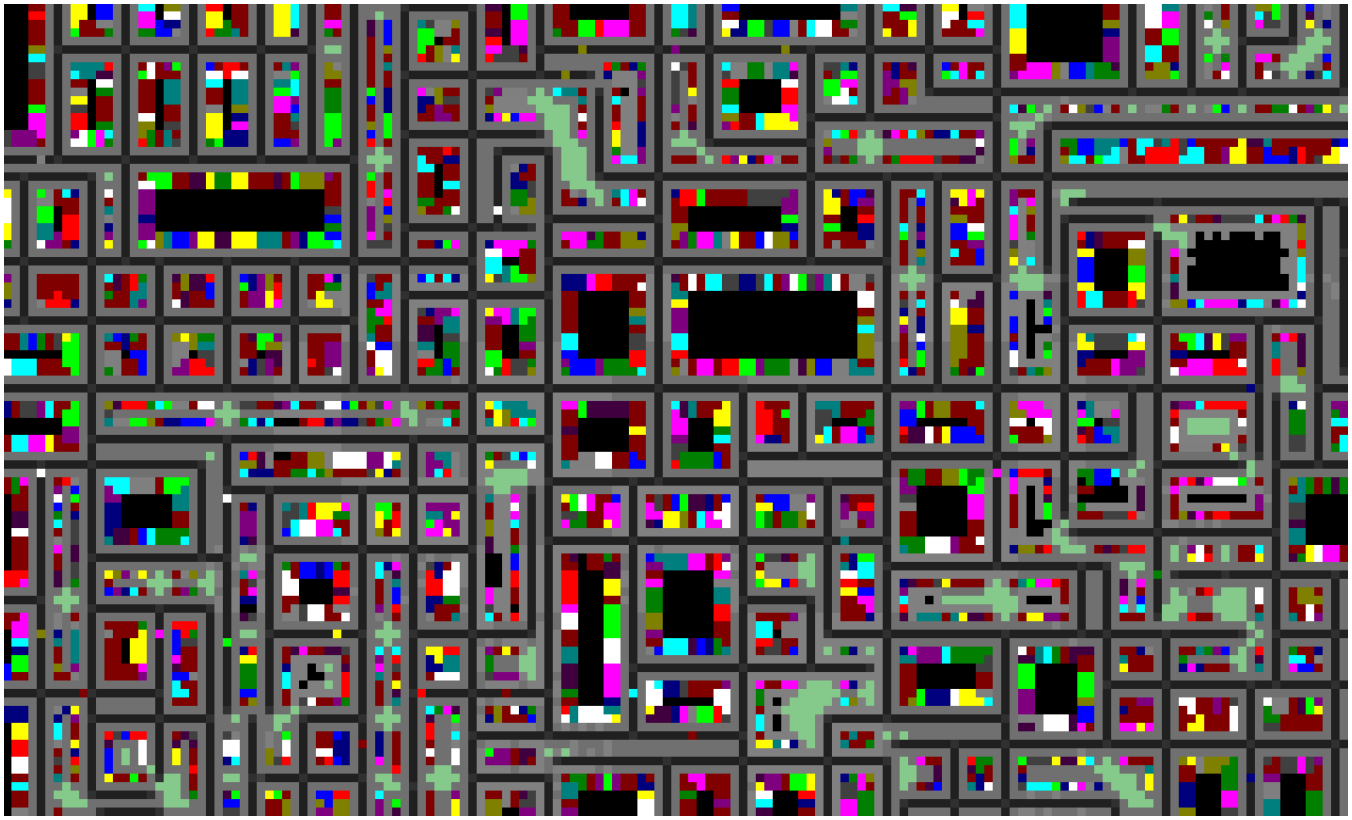

Figure 5: A city after its City Growth stage on a medium MFM grid.

visibility, a routing system that uses street signs and intersection canalization is rather effective at helping traffic get to where it needs to go. These increases in efficiency may seem small, but over time more people are spending far less time traveling than people who are hopelessly lost otherwise.

Unfortunately, as stated earlier, this routing method does not seem to be applicable in a real city because of the large amount of information that people are able to access at any given time. However, considering that these elements are rather simple, this method works well with the limited resources given by the MFM. The small bit fields used on the MFM also make these routing systems less effective, considering that another 20 bits in each atom would allow for the sidewalk maps to extend from seeing 3 blocks away to seeing 7 blocks away. These limitations are what developers of the MFM must be wary of and ready to work around.

We can show that all of our design constraints listed earlier have all been satisfied by Intersection-Canalization routing. Clearly, the first consideration is satisfied by sidewalk mapping. This routing method is fast, because every car is routed in $O(n)$ time where n is the number of building types (and currently, there cannot be more than 24 building types given the atom body size limitations of the MFM). It is accurate, becase 98.2% of cars arrive safely at their destinations. Is is simple, since all routing decisions are made using the simple algorithm described by Algorithm1. Finally, it is flexible, since the unused data inside of each car (67 bits if we do not track gas usage, or 59 bits if we do) is large enough to be considered a general-use packet for any kind of data transmission. Finally, the use of self-organizing streets helps increase the robustness of the city past what we would hope to get out of a self-assembled city (Misteli, 2001).

Considering that this system takes all of our design considerations into account, we can see that this is an effective routing algorithm for any kind of information. It can be seen that a city can be construced in a way that allows traffic to carry data across tile boundaries, effectively allowing this routing system to route data between other computational parts of the MFM(for instance, this system can be placed on one end of a DEMON HORDE SORT(Ackley and Small, 2014) to route sorted data to another part of the machine).

Future Work

In the future on a new build of the MFM, it would be interesting to study the effects that larger atom bodies have on these routing methods. As a whole, larger atom bodies on the MFM would allow for even more complicated routing methods. The adaptive traffic lights studied in (Tubaishat et al., 2007) could theoretically be implemented, as well as a host of other traffic simulations. This study shows that

small bit fields work, but leaves us wondering if more state could improve these routing methods.

The MFM is still a young architecture which is very much in development. However, it can be seen by this and other projects that it is a promising architecture that should be used (maybe after certain parameters like atom sizes are tweaked) for further research. I encourage anyone who wishes to use a cellular automata in the future to try their hand at using the MFM, as it scales well and can handle any general-purpose cellular automata.

On top of the routing research done in this paper, this project shows that the MFM (or, rather, the MFM simulator) is able to produce visually stunning results. In the future it would be interesting to create other visual projects on this architecture.

See Also

Each routing algorithm is exhibited in the following videos:

Random routing:

Sidewalk-Only routing:

Intersection-Canalization routing:

The source code for this project is provided as an open-source fork of the MOVABLE FEAST MACHINE V2 codebase, available at:

http://github.com/sixstring982/MFMv2-city

Acknowledgements

I would like to thank UNM Professor Dave Ackley for allowing me to aide in the research of the Movable Feast Machine over the past year. I would also like to thank Google for funding this research.

References

Ackley, D. H., Cannon, D. C., and Williams, L. R. (2013). A movable architecture for robust spatial computing. *The Computer Journal*, 56.12:1450–468.

Ackley, D. H. and Small, T. R. (2014). Indefinitely scalable computing = artificial life engineering. In Sayama, H., Rieffel, J., Risi, S., Doursat, R., and Lipson, H., editors, *Artificial Life XIV*, pages 606–613. MIT Press, Cambridge, MA.

Brockfeld, E., Barlovic, R., Schadschneider, A., and Schreckenberg, M. (2001). Optimizing traffic lights in a cellular automaton model for city traffic. *Physical Review E*, 64(5):056132.

Clarke, G. u. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581.

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., and Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research society*, pages 512–522.

Gershenson, C. and Rosenblueth, D. A. (2009). Modeling self-organizing traffic lights with elementary cellular automata. *arXiv preprint arXiv:0907.1925*.

Hsu, M. K., Huang, Y., and Swanson, S. (2010). Grocery store image, travel distance, satisfaction and behavioral intentions: Evidence from a midwest college town. *International Journal of Retail & Distribution Management*, 38(2):115–132.

Lämmer, S. and Helbing, D. (2008). Self-control of traffic lights and vehicle flows in urban road networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(04):P04019.

Lu, Q. and Dessouky, M. (2004). An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, 38(4):503–514.

Misteli, T. (2001). The concept of self-organization in cellular architecture. *The Journal of Cell Biology*, 155.2:181–86.

Nagel, K. and Schreckenberg, M. (1992). A cellular automaton model for freeway traffic. *Journal de physique I*, 2(12):2221–2229.

Nagel, K., Wolf, D. E., Wagner, P., and Simon, P. (1998). Two-lane traffic rules for cellular automata: A systematic approach. *Physical Review E*, 58(2):1425.

Parker, L. (2014). Not just playing around anymore: Games for change uses video games for social projects. *The New York Times, April 22 2014*.

Tubaishat, M., Shang, Y., and Shi, H. (2007). Adaptive traffic light control with wireless sensor networks. In *Proceedings of IEEE Consumer Communications and Networking Conference*, pages 187–191.

Wiering, M., Vreeken, J., Van Veenen, J., and Koopman, A. (2004). Simulation and optimization of traffic in a city. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 453–458. IEEE.