Improved Bounds for the Firing Synchronization Problem

A. SETTLE
DePaul University & University of Chicago, USA

J. SIMON
University of Chicago, USA

Abstract

In this paper we present improved bounds on the complexity of solutions to the firing synchronization problem. In the firing synchronization problem we consider a one-dimensional array of n identical finite automata. Initially all automata are in the same state except for one automaton which is designated as the initiator for the synchronization. Our results hold for the original problem, where the initiator may be located at either endpoint, and for the variant where any one of the automata may be the initiator, called the generalized problem. In both cases, the goal is to define the set of states and transition rules for the automata so that all machines enter a special fire state simultaneously and for the first time during the final round of the computation. In our work we improve the construction for the best known minimal-time solution to the generalized problem by reducing the number of states needed and give non-minimal time solutions to the original and generalized problem which use fewer states than the corresponding minimal-time solutions.

Keywords: cellular automata, firing squad synchronization problem

1 Introduction

In the firing synchronization problem we consider a one-dimensional array of n identical finite automata. Initially all automata are in the same state except for one automaton, which is designated as the initiator for the synchronization. The machines operate in lock-step, and the transitions of each automaton depend on the states of the automaton and its neighbors. The goal is to define the set of states and transition rules for the automata so that all machines enter a special fire state for the first time and simultaneously during the final round of the computation.

Synchronizing a set of processes is an important problem in distributed algorithms and the firing synchronization problem is one of the simplest and oldest formalizations of this problem. By studying this fundamental and elegant question we hope to gain insight into other such problems and develop techniques and intuitions that will be useful for generalizations of the problem. For example, solutions to more general versions of the firing synchronization problem, in which the underlying network is an undirected or strongly-connected directed graph, work by reducing the graph to simpler structures which are synchronized by solutions to the one-dimensional problem [Kob78, NH81, ELW97, OW95].

Two obvious criteria for ranking solutions are the speed of the solution, namely the time needed to synchronize, and the complexity of the solution, measured by the number of states of the automaton.

It is easy to show that in the original problem an array of n automata cannot synchronize before time step $2n - 2$ [Moo64]. This is the minimal amount of time for the initiator to send a message to the far end automaton and get a message back. A minimal-time solution is a set of states and transition rules for which synchronization occurs after exactly $2n - 2$ time steps, whereas a non-minimal time solution is one where synchronization takes more than $2n - 2$ time steps. The same notions can also be defined for the generalized version of the problem, and there is a similar bound on the number of time steps necessary for synchronization.

In this paper we improve the bounds on the complexity of solutions to the firing synchronization problem. We give a 9-state minimal-time automaton for a generalized version of the problem. This improves on the best previously known construction, an automaton using 10 states, which appeared in a paper by Szwerinski [Szw82]. We give a 6-state non-minimal time automaton for the original problem where the initiator may be located at either endpoint. This automaton uses two fewer states than the best known minimal-time automaton for the same problem [Bal67]. We also present a 7-state non-minimal time solution to the generalized problem which uses 2 fewer states than the best known minimal-time automaton mentioned above. Both of our non-minimal time automata are based on a 6-state solution to a restricted version of the problem produced by Mazoyer [Maz87]. We also have a proof of correctness for each of our non-minimal time automata.

1.1 Previous Work

The firing synchronization problem has a long history. Initially proposed by Myhill, the first published solution is by McCarthy and Minsky [Min72]. Their automaton uses a divide and conquer algorithm and takes $3n$ steps to synchronize. The first minimal-time automaton for the original problem was produced by Goto, who gave a solution with over 1000 states in 1962 [Got62], and work in the area quickly focused on finding minimal-time solutions using fewer states. In 1966 Waksman [Wak66] gave a 16-state minimal-time solution, and Balzer [Bal67] independently produced an 8-state solution using the same ideas. Balzer also showed, using a heuristic search algorithm, that there is no 4-state minimal-time solution to the original problem. (See also Sanders' paper [San94] for a recent result confirming the bound). In 1987 Mazoyer [Maz87] produced a 6-state solution to a restricted version of the problem in which the initiator is located at the left endpoint of the array.

Mazoyer suggested that all solutions with few states must necessarily be minimal-time, a conjecture based on the idea that the simplest solution will naturally be the fastest. Yunès [Yun94] contested the conjecture[1] in 1994 by giving an implementation of McCarthy and Minsky's solution that requires only 13 states and time $t(n) = 3n \pm \Theta_n \log n + C$, where $0 \leq \Theta_n < 1$ and by producing a 7-state solution that uses time $t(n) = 3n \pm 2\Theta_n \log n + C$, where $0 \leq \Theta_n < 1$. Both automata solve the restricted version of the original problem which requires that the initiator be located at the left endpoint.

Moore and Langdon introduced a generalization of the original problem in 1968 [ML68]. They also considered a one-dimensional array of finite automata but allowed the initiator to be located anywhere in the array. In their paper, Moore and Langdon gave a 17-state minimal-time solution for the generalized problem. Varshavsky, Marakkovsky and Peschansky [VMP70] improved this result, producing a 10-state minimal-time solution.

Further work on the generalized problem was done in 1982 by Szwerinski [Szw82]. Szwerinski considered symmetric solutions. A symmetric solution is one in which an automaton cannot distinguish between its left and right neighbors. Szwerinski gave a 10-state, symmetric, minimal-time solution.

To the best of our knowledge, non-minimal time solutions to the generalized problem have not been studied previously.

1.2 Lower versus Upper Bounds

Despite its long history, there are many important open problems remaining for the firing synchronization problem. One of the most fundamental is determining precisely how many states an automaton solving the problem requires.

Balzer has shown that no 4-state minimal-time automaton for the restricted version of the original problem exists. In each variant of the problem this leaves a gap between the lower bound and the best known minimal-time solutions. For the unrestricted original problem this gap is 4 states. Any lower bound for the original problem also applies to the generalized problem since the original problem must be solved as a subcase. Thus the gap for the generalized problem is 6 states.

Work on non-minimal time solutions has been even more limited. The only known lower bound for non-minimal automata is a 3-state bound on solutions to the original firing synchronization problem [SS98]. This leaves a gap of 4 states between the best known solution to the original problem and the lower bound. As stated before there were no known non-minimal time solutions to the generalized problem prior to this work.

[1] Recall that the exact size of the optimal minimal-time automaton is not known.

1.3 Our Contributions

In Section 3 we present a 9-state minimal-time symmetric solution to the generalized firing synchronization problem. Its transition function may be found in Appendix A. The automaton contains within it an 8-state symmetric solution to the original problem. The 9-state automaton has the fewest states of any known minimal-time solution to the generalized problem.

We present in Section 4.1 a 6-state non-minimal time solution to the original problem which allows the initiator to be located at either the left or the right endpoint of the array. The transition function for the automaton may be in found in Appendix A. This automaton has 2 fewer states than Balzer's 8-state minimal-time automaton [Bal67]. We also have a proof of correctness for the solution which shows that for any $n \in \mathcal{N}^{\geq 2}$ the solution synchronizes a one-dimensional array of n automata in $2n-1$ steps if the initiator is located at the left endpoint of the array or $3n+1$ time steps if the initiator is located at the right endpoint. The details of the proof are not given here but may be found in the technical report [SS97].

Finally in Section 4.2 we present a 7-state non-minimal time solution to the generalized problem mentioned above, where the initiator can be anywhere in the array. The transition function for the automaton may also be found in Appendix A. This automaton has the fewest states of any known solution to this problem, and requires 2 fewer states than our minimal-time automaton. We also have a proof of correctness for the 7-state solution which shows that the solution synchronizes a one-dimensional array of n automata with initiator located in position k of the array in $2n-2+k$ time steps for any $n \in \mathcal{N}^{\geq 2}$. Again the details of the proof are omitted but may be found in the technical report [SS97].

Few researchers in this area have provided correctness proofs for their automata. Indeed, as far as we know, prior to this work only Balzer [Bal67] and Mazoyer [Maz87] have published proofs of correctness.

Our work provides additional evidence that Mazoyer's conjecture does not hold by giving non-minimal time solutions to both the original and generalized versions of the firing synchronization problem which require fewer states than the best known minimal-time solutions. Indeed optimal non-minimal time solutions may use even fewer states than our constructions, as our automata are built on top of minimal-time solutions to restricted versions of the problem.

2 The Firing Synchronization Problem

The firing synchronization problem, sometimes also called the firing squad problem, is a classical problem of synchronization. Consider a one-dimensional array of n finite automata in which all automata are identical except the ones on either end of the array. The machines work synchronously, and the state of an automaton at time t only depends on its and its neighbors' states at time $t-1$. In round 0 of the computation all automata are in a special quiescent state except for one automaton which is designated as the initiator for the synchronization. The problem is to define the set of states and transition rules for the automata so that all machines enter a special fire state simultaneously and for the first time at some time $t(n)$.

The transition function for each automaton can be given as a set of 4-tuples. The 4-tuple (X,Y,Z,W) represents the rule that an automaton currently in state Y, with left neighbor in state X and right neighbor in state Z will enter state W at the next time step. We will denote this by XYZ $\rightarrow$ W. By definition automata solving the firing synchronization problem are deterministic so that there is at most one tuple (X,Y,Z,W) for any triple of states X,Y,Z.

It is easy to show that $t(n) \geq 2n - 2$ [Moo64]. This is the minimal amount of time for the initiator to send a message to the far end automaton and get a message back.

A minimal-time solution is a set of states and transition rules for which $t(n) = 2n - 2$, whereas a non-minimal time solution is one for which $t(n) > 2n - 2$. An N-state solution of the problem is one in which each automaton has N states, including the quiescent and fire states.

A symmetric automaton is one which has a symmetric transition function, that is, whenever a transition XYZ $\rightarrow$ W is defined, the transition ZYX $\rightarrow$ W must also be defined. This means that the automata cannot distinguish their left and right neighbors.

A generalization of the original problem introduced by Moore and Langdon [ML68] allows the initiator to be located anywhere in the one-dimensional array of finite automata. Let k denote the position of the initiator in the array, where $1 \leq k \leq n$, and let $m = \min \{k-1, n-k\}$. Moore and Langdon showed that $2n-m-2$ is the minimal firing time for the generalized problem.

3 The Minimal-Time Solution

In this section we describe the 9-state minimal-time solution to the generalized firing synchronization problem. The 9-state automaton is a modification of Szwerinski's 10-state solution.

The strategy for Szwerinski's automaton, like all other known solutions to the generalized problem [ML68, VMP70], is to reduce the synchronization of the generalized problem to the original problem. Once this has been completed, the synchronization is finished by a solution to the original problem contained within the transition function for the generalized solution. For this reason, Szwerinski's solution works in two phases, the first of which accomplishes the reduction to the original problem, and the second which completes the synchronization using the underlying original solution.

Szwerinski's 10-state automaton contains an 8-state solution to the original problem and uses two additional states for the first phase of the synchronization. The 9-state automaton also contains the 8-state original solution, but uses only one additional state for the first phase.

In the remainder of the section, we describe at a high level how the 9-state automaton works. We then give a detailed explanation of the underlying 8-state solution to the original problem and finally explain how the two phases work in the 9-state automaton.

3.1 A High-Level Description

In the 9-state minimal-time solution, the line is repeatedly divided into halves as new initiators are placed in the center of each of the intervals. The simulation ends when all automata become initiators and fire at the next time step.

The transition function for the 9-state automaton is given in Appendix A. It can be seen from the transition function that there are several states which propagate toward neighboring automata. We call these states signals, since their purpose is to carry information from one part of the array to another. Other states remain stationary until they come into contact with certain signals. We call these states markers. They act as placeholders indicating significant positions in the array, such as the center of the line.

In order to understand how the division of the array is performed, consider what happens when the initiator is located at either end. A simulation for this case can be found in Appendix B. The time steps given below refer to the simulation in that section.

This case is simply the original problem and is handled by the underlying 8-state automaton. The initiator sends out a signal which produces a second initiator when it reaches the opposite end of the line. In the sample simulation this occurs between time steps 0 and 16. When this wake-up signal is reflected back by the new initiator, it intersects with markers created in the wake of the first signal and produces a third initiator (or pair of initiators depending on the parity of the original line) located at the center of the array. This occurs at time step 24 of the sample simulation. This division of the line continues until every other automaton is an initiator, which occurs at time step 30. At the next time step in the simulation every automaton becomes an initiator, and at the following time step all automata fire.

In the case where the initiator is located somewhere in the middle of the array, the goal is to reduce the problem to the original problem. To achieve this reduction, a new initiator is produced at the center of the array at time $n - m + \lfloor \frac{n}{2} \rfloor$. The simulation then continues from that point as if the first initiator had been located at one of the endpoints. An extra state is used to achieve this first subdivision, but after the central initiator is created, the remainder of the simulation is handled by the subset of states corresponding to the 8-state solution and the extra state does not appear.

3.2 The Underlying 8-State Solution to the Original Problem

In order to explain in some detail how the 9-state solution works, we first present the 8-state solution to the original problem. Recall that the original problem requires that the first initiator be located at one of the endpoints of the array of automata. The 8-state solution is derived from the 9-state solution to the generalized problem by deleting all occurrences of the state D. The time steps given below refer to the simulation found in Appendix B. The solution works as follows.

The first initiator in state G sends out an A-signal to the other end of the line. In the simulation the A-signal is produced at time step 1. The A-signal moves at a rate of one automaton per time step. As the A-signal advances away from an automaton, it leaves the automaton in one of two states, either R or P. An R is produced at all even time steps and a P at all odd time steps. Thus the parity of the line segment the A-signal crosses can be determined by the state appearing behind the A-signal once it reaches the end of the line.

The R moves back in the direction from which the A-signal came at the rate of one automaton per time step. The first R-signal is produced at time step 2 of the sample simulation. When the R-signal collides with the initiator at the other end it produces a B-marker. This occurs in the simulation at time step 3. The new B-marker moves away from the initiator one position each time it encounters a new R-signal. For example, the first B-marker advances at time step 6 of the simulation. This first marker will be the one that will mark the center(s) where the next initiator(s) should be produced. In order to mark the $\frac{1}{4}$, $\frac{1}{8}$, $\ldots$, positions in the array, where the next initiator(s) need to be placed, additional B-markers need to be produced. This is done by allowing the R-signal to continue past a B-marker every other time a B-marker advances. The R-signal can then produce and/or advance other B-marker(s). The state of the automaton behind the B-marker determines whether the R-signal advances. If the state of the automaton behind the B-marker is a P, then the R-signal will regenerate behind the B-marker after advancing it. An example of this can be seen at time steps 11, 12 and 13 of the simulation. On the other hand, if the B-marker is followed by an automaton in state Z, the R-signal will vanish after moving the B-marker forward. This case can be seen at time steps 14 and 15.

As the A-signal hits the end of the array, it is reflected back. In the sample simulation this occurs at time steps 15, 16 and 17. Depending on the parity of the array, one of two state configurations will be produced behind the A-signal as it advances. If the line is of odd length, the A-signal will be followed at alternating time steps by an R or P, as with the first A-signal. If the array has even length, then the A will be followed by a Z which is alternately followed by an R or P. The sample simulation has 17 automata so that the former case holds.

By the time the A-signal is reflected back, it has sent enough R-signals to bring the B-marker to the middle of the line. This is because the A-signal produces an R-signal every other time step, creating half as many R-signals as the length of the array. Because the leading B-marker moves one position each time it encounters one of these R-signals, it will have moved to the center of the array once all of the R-signals reach it. This happens before the reflected A-signal can reach the B-marker.

When the A-signal reaches the B-marker it produces the new initiator(s). In the simulation this occurs at time step 24. A single new initiator is produced if the line has odd length, since in that case there is a middle point of the array. This is true if the A-signal reaches the B-marker with the automaton behind it in state P. If the line length has even parity then two new initiators need to be produced since there are two central positions in the array. This occurs when the A-signal reaches the B-marker with the automaton behind it in state Z. Again, because there are 17 automata in the simulation, the former case holds.

The new initiator(s) now begins to recursively subdivide the array. A-signals are sent out toward each end of the array. These will intersect the remaining B-markers produced in the wake of the first A-signal and the reflected A-signal to produce the initiators at the quarter positions. This process continues until every other automaton is an initiator. At that point all automata become initiators and then fire at the next time step.

3.3 The 9-state Automaton

The 9-state solution to the generalized problem works in a manner similar to the 8-state automaton. The additional state D is used as the state for the first initiator. If this initiator is at the end of the line, it sends an A-signal toward the other end of the array and enters state G. The rest of the algorithm is then the same as the 8-state automaton described above.

In the case where the first initiator is located somewhere in the middle of the array, the goal is to reduce the synchronization task to the original problem. The first initiator begins this process by sending out A-signals in both directions. The R-signals produced in the wake of the A-signals meet at the initiator and disappear. The A-signals will create new initiators as they reach the end of the line and are reflected back toward the middle. If the array is of odd length and the initiator is located at the center point, then the A-signals will meet back at the original initiator, putting it into state G. If the initiator is not located at the center point of the array, then the A-signal sent to the closer end returns to the initiator first. If the length of the shorter segment is even, a D-marker is produced when the A-signal reaches the initiator. If the shorter segment has odd parity, then the A-signal creates a B-marker when it reaches the initiator. These markers now advance in response to the R-signals sent by the A-signal on the opposite side and move to the center of the array. There they are met by the reflected A-signal and create initiator(s) in state G. Whether a single initiator or two initiators are produced is determined by the parity of both the short and long line segments, which is encoded both by the marker state and the state of the automaton behind the A-signal.

Once the central initiator(s) are created, the remainder of the simulation is performed by the 8-state automaton, as described in the previous subsection. The state D does not appear after this point.

4 The Non-minimal Time Automata

4.1 A 6-state Automaton for the Original Problem

The 6-state automaton is based on Mazoyer's 6-state solution to the restricted version of the original firing synchronization problem. Recall that Mazoyer's minimal-time automaton requires the initiator to be located at the left endpoint of the array. Mazoyer's solution works by dividing the line of automata into unequal parts, one of length $\frac{2}{3}n$ and the other of length $\frac{1}{3}n$. An initiator is placed at the left endpoint of the shorter segment, and each segment is then recursively subdivided. After every automaton becomes an initiator, the automata fire and the synchronization ends. For a detailed description of that solution see Mazoyer's paper [Maz87].

Unlike Mazoyer's solution, the initial configuration for our 6-state non-minimal time automaton allows the initiator to be located at either the left or right endpoint of the array. In either case the goal of the non-minimal time automaton is to produce the initial configuration necessary for Mazoyer's solution. The synchronization of the array is then completed by the minimal-time automaton.

4.1.1 The Description of the Solution

The behavior of our 6-state automaton is as follows. The state B is used as the state for the first initiator. If the initiator is located at the left endpoint in the initial configuration, the automaton simply enters the state G at the next time step. This puts the array in the configuration necessary for the minimal-time automaton, which then synchronizes the line. The entire process takes one additional step beyond the time for the minimal-time synchronization, and the line is synchronized in time $2n - 1$.

If the initiator is located at the right endpoint when the synchronization begins, a signal is sent toward the left endpoint. The purpose of this signal is to produce an initiator in state G at the left end of the array, leaving the rest of the automata in the quiescent state L as the signal passes. This puts the array in the configuration necessary for the minimal-time solution, which then completes the synchronization.

The signal which produces the initiator at the left endpoint consists of four states, AACB. The B initiator enters state A at time step 1, and the A then advances left, producing the rest of the signal behind it during time steps 2 through 4. The signal then moves at a rate of one automaton per time step toward the left. As the signal moves, the automata behind it are once again put into the quiescent state L. When the signal reaches the left end of the array, the signal collapses, leaving only the last two states in the signal. At the next time step the G initiator is produced and the minimal-time synchronization takes place.

One step is necessary to produce the lead state A in the signal. Another n time steps are required for the A to reach the left endpoint. It then takes an additional two time steps for the first two states of the signal to vanish and create the G initiator. The minimal-time automaton then finishes the synchronization. Thus the whole process takes time $1 + n + 2 + (2n - 2) = 3n + 1$.

The transition function for the 6-state automaton may be found in Appendix A. The proof of correctness for the automaton proceeds by induction on the time step of the synchronization. The details are omitted but may be found in the technical report [SS97].

4.2 The 7-State Solution to the Generalized Problem

The 7-state automaton, like the 6-state solution to the original problem, is based on Mazoyer's 6-state minimal-time solution, which was briefly described in the previous section. It allows the first initiator to be located anywhere in the array and works by sending a signal from the first initiator back toward the left endpoint. When the signal reaches the end of the line, it transforms into the initiator for Mazoyer's 6-state solution, and the minimal-time synchronization begins.

In order to allow the first initiator to be located anywhere in the array, a new state D is added to Mazoyer's automaton. D is used both for the first initiator state and as the state for the signal that moves leftward. This results in the D migrating across the line of automata until it reaches the end. Once the D signal reaches the left endpoint, it puts the leftmost automaton in state G, the initiator state for Mazoyer's automaton. The synchronization is then completed by the 6-state minimal-time solution.

If the first initiator is located in position k of the array, it takes $k - 1$ steps for the D signal to reach the left endpoint. At the next time step the D transforms into a G and the minimal-time synchronization begins. This means that the entire synchronization takes time $2n - 2 + k$ time steps.

The transition function for the 7-state automaton can be found in Appendix A. The proof of correctness for the automaton proceeds by induction on the time step of the synchronization. The details are omitted but may be found in the technical report [SS97].

5 Conclusion

In this paper we presented improved bounds on the complexity of one-dimensional variants of the firing synchronization problem. We gave a 9-state minimal-time automaton for a generalized version of the problem, which has the fewest states used by any minimal-time automaton solving that variant. We gave a 6-state non-minimal time automaton for the original problem which allows the initiator to be located at either endpoint. We also presented a 7-state non-minimal time solution to the generalized problem, the only known non-minimal time solution for the generalized problem. We also have a proof that each of the non-minimal time automata correctly solve the firing synchronization problem.

This work narrows the gaps between the upper and lower bounds on the number of states required for an automaton solving the firing synchronization problem. The 6-state non-minimal time automaton for the unrestricted original problem presented here uses 2 fewer states than the best known minimal-time automaton solving the same problem and uses only 3 states more than the lower bound on non-minimal time solutions to the problem.

Progress is also made in the generalized case. The lower bound for minimal-time solutions to the original problem applies to the generalized problem. We give a 9-state minimal-time solution and a 7-state non-minimal time automaton for the generalized problem. In this case, the minimal-time solution uses 5 states more than the lower bound and the non-minimal time solution uses only 4 states more than the lower bound on non-minimal time automata.

6 Acknowledgments

We are indebted to Sophie Laplante for many productive discussions about these results and for her suggestions on improvements to drafts of the work. We would also like to thank André Berthiaume and Marcus Schäfer for their comments on earlier drafts of this paper.

References

[Bal67] R. Balzer. An 8-state minimal time solution to the firing squad synchronization problem. *Information and Control*, 10:22–42, 1967.

[ELW97] S. Even, A. Litman, and P. Winkler. Computing with snakes in directed networks of automata. *Journal of Algorithms*, 24(1):158–170, 1997.

[Got62] E. Goto. A minimum time solution of the firing squad problem. Course Notes for Applied Mathematics 298, Harvard University, 1962.

[Kob78] K. Kobayashi. The firing squad synchronization problem for a class of polyautomata networks. *Journal of Computer and System Sciences*, 17(3):300–318, 1978.

[Maz87] J. Mazoyer. A six-state minimal time solution to the firing synchronization problem. *Theoretical Computer Science*, 50:183–238, 1987.

[Min72] M. Minsky. Computation*: Finite and Infinite Machines*. Prentice Hall, 1972.

[ML68] F. R. Moore and G. G. Langdon. A generalized firing squad problem. *Information and Control*, 12:212–220, 1968.

[Moo64] E. F. Moore. The firing squad synchronization problem. In *Sequential Machines - Selected Papers*, pages 213–214. Addison-Wesley, 1964.

[NH81] Y. Nishitani and N. Honda. The firing squad synchronization problem for graphs. *Theoretical Computer Science*, 14(1):39–61, 1981.

[OW95] R. Ostrovsky and D. Wilkerson. Faster computation on directed networks of automata. In *Proceedings of 14th Annual ACM Symposium on Principles of Distributed Computing*, pages 38–46, 1995.

[San94] P. Sanders. Massively parallel search for transition-tables of polyautomata. In C. Jesshope, V. Jossifov, and W. Wilhelmi, editors, *Proceedings of the VI International Workshop on Parallel Processing by Cellular Automata and Arrays*, pages 99–108. Akademie, 1994.

[Set97] A. Settle. A symmetric nine-state automaton for the generalized firing synchronization problem. Technical Report 97–03, University of Chicago Department of Computer Science, 1997.

[SS97] A. Settle and J. Simon. Non-minimal time solutions for the firing synchronization problem. Technical Report 97–08, University of Chicago Department of Computer Science, 1997.

[SS98] A. Settle and J. Simon. A non-minimal time lower bound for the firing synchronization problem. Unpublished manuscript, 1998.

[Szw82] H. Szwerinski. Time-optimal solution of the firing-squad-synchronization-problem for n-dimensional rectangles with the general at an arbitrary position. *Theoretical Computer Science*, 19:305–320, 1982.

[VMP70] V. I. Varshavsky, V. B. Marakhovsky, and V. A. Peschansky. Synchronization of interacting automata. *Mathematical Systems Theory*, 4:212–230, 1970.

[Wak66] A. Waksman. An optimum solution to the firing squad synchronization problem. *Information and Control*, 9:66–78, 1966.

[Yun94] J. B. Yunès. Seven-state solutions to the firing squad synchronization problem. *Theoretical Computer Science*, 127:313–332, 1994.

A The Transition Functions

Table 1 shows the transition function for the 9-state automaton. The state of an automaton at the next time step can be found by looking at the entry in the column corresponding to the automaton's present state and the row corresponding to the states of its neighbors. Since the automaton is symmetric, the orientation of the neighbors is irrelevant. The $\star$ is used to indicate the end of the array. In order to obtain the 8-state automaton that solves the original problem, the column corresponding to D must be removed and all occurrences of D deleted in the remaining table.

Table 2 and Table 3 give the transition functions for the non-minimal time automata. The state of an automaton at the next time step can be found by looking at the table corresponding to the automaton's present state. The state that the automaton should enter at the next time step is the one in the row and column corresponding to the states of its left and right neighbors respectively. As in Table 1, $\star$ is used to denote the end of the array.

	present state							
neighbors' states	Z	A	B	D	R	P	Q	G
Z–Z	Z	Z	B	R			Q	G
Z–A	A	Z	G		A		D	
Z–B	Z	G	B	P		P		
Z–D	A	R	D	G		P	Z	
Z–R	R	P	P	D	Q	R	Z	G
Z–P	Z	R	B	D	Q			
Z–Q	Z	R	B	D	Q	R		
Z–G	A	R			B	A		G
Z–$\star$	Z			G				G
A–A		G		G	Q		G	G
A–B	A	G	G		G	P		
A–D					D			
A–R	P			G			A	
A–P	R		G	G	Q		D	
A–Q	A	Z	G	B	A	P		
A–G	R	G	G		B			G
A–$\star$	G	G						G
B–B	Z			P		P	P	G
B–D	Z			G		P		
B–R	R	P	P	P	B	R	Z	G
B–P	Z	R		R			Q	

	present state							
neighbors' states	Z	A	B	D	R	P	Q	G
B–Q	Z			P	B	R	P	
B–G	A	R	B			A	A	G
B–$\star$								G
D–R	R	Q		G	Z		Z	
D–P				G	R		Q	
D–Q	Z			G	B	P		
D–G	B		B		B			
D–$\star$	G							
R–R		P		D	Z		D	G
R–P		R	Q	D	Q		Z	A
R–Q	R	P		D		R	Z	G
R–G		R	B		A		A	G
R–$\star$								G
P–P							Q	A
P–Q	Z	R					Z	
P–G			B		A		A	A
P–$\star$								A
Q–Q				Q	Q			A
Q–G	A	R	B			A	A	G
G–G	G		G		G			F
G–$\star$	G							F

Table 1: The Transition Function for the 9-state Automaton

A	A	B	C	G	L	⋆
A	A	B	C	B	A	F
B		G	C	C	G	C
C	A				A	
G			C	C		C
L	A	L	G			
⋆	F		G			

B	A	B	C	G	L	⋆
A	B	B	L		G	
B	A	B	C	B	G	
C	A			L	L	L
G	C		B	G	C	G
L	G	B	L	B		

C	A	B	C	G	L	⋆
A		B		B	B	B
B			C	G	C	G
C	A	B	C	B	C	
G		B		B	B	B
L	A	G	C	G	C	

D	L	⋆
L	L	L
⋆	G	

L	A	B	C	D	G	L	⋆
A	L	L	L		C	G	C
B	L	L	L		L	L	L
C	L	L	L		G	A	G
D						L	L
G	L	L	L		A	C	A
L		L	L	D	L	L	L
⋆				D		L	

G	A	B	C	G	L	⋆
A		G	G		B	
B		G	G	G	B	G
C		G	G	A		
G		G	G	F	B	F
L	G	G	G			
⋆		G	G	F	A	

Table 2: The Transition Function for the 7-state Automaton

A	A	B	C	G	L	⋆
A	A	B	C		A	F
B		G	C	C	G	C
C	A				A	
G			C	C		C
L	A	L	G	A		G
⋆	F		G			

B	A	B	C	G	L	⋆
A	B	B	L		G	
B	A	B	C	B	G	
C	A			L	L	L
G	C		B	G	C	G
L	G	B	L	B		A
⋆	B		C	L	G	

C	A	B	C	G	L	⋆
A		B		B	B	B
B		B	C	G	C	G
C	A	B	C	B	C	
G		B		B	B	B
L	A	G	C	G	C	
⋆		G				

L	A	B	C	G	L	⋆
A	L	L	L	C	G	C
B	L	L	L	L	L	L
C	L	L	L	G	A	G
G	L	L	L	A	C	A
L	A	L	L	L	L	L
⋆	B	L		B	L	

G	A	B	C	G	L	⋆
A		G	G		B	C
B		G	G	G	B	G
C		G	G	A	A	A
G		G	G	F	B	F
L	G	G	G			L
⋆		G	G	F	A	

Table 3: The Transition Function for the 6-state Automaton

B Sample Simulation

The following is a simulation of the underlying 8-state solution to the original problem contained within the 9-state automaton. The length of the array is 17 and the first initiator is located in position 1. The synchronization begins at time step 0 and firing occurs at time step 32. The current time step is given to the left of the array.

0 : G Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z
1 : G A Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z
2 : G *R* A Z Z Z Z Z Z Z Z Z Z Z Z Z Z
3 : G **B** P A Z Z Z Z Z Z Z Z Z Z Z Z Z
4 : G **B** P *R* A Z Z Z Z Z Z Z Z Z Z Z Z
5 : G **B** *R* Q P A Z Z Z Z Z Z Z Z Z Z Z
6 : G **B** **B** Z P *R* A Z Z Z Z Z Z Z Z Z Z
7 : G **B** **B** Z *R* Q P A Z Z Z Z Z Z Z Z Z
8 : G **B** **B** *R* Q Z P *R* A Z Z Z Z Z Z Z Z
9 : G **B** P **B** Z Z *R* Q P A Z Z Z Z Z Z Z
10 : G **B** P **B** Z *R* Q Z P *R* A Z Z Z Z Z Z
11 : G **B** P **B** *R* Q Z Z *R* Q P A Z Z Z Z Z
12 : G **B** P Q **B** Z Z *R* Q Z P *R* A Z Z Z Z
13 : G **B** *R* Q **B** Z *R* Q Z Z *R* Q P A Z Z Z
14 : G **B** **B** Z **B** *R* Q Z Z *R* Q Z P *R* A Z Z
15 : G **B** **B** Z P **B** Z Z *R* Q Z Z *R* Q P A Z
16 : G **B** **B** Z P **B** Z *R* Q Z Z *R* Q Z P *R* G
17 : G **B** **B** Z P **B** *R* Q Z Z *R* Q Z Z *R* A G
18 : G **B** **B** Z P Q **B** Z Z *R* Q Z Z *R* A *R* G
19 : G **B** **B** Z *R* Q **B** Z *R* Q Z Z *R* A P **B** G
20 : G **B** **B** *R* Q Z **B** *R* Q Z Z *R* A *R* P **B** G
21 : G **B** P **B** Z Z P **B** Z Z *R* A P Q *R* **B** G
22 : G **B** P **B** Z Z P **B** Z *R* A *R* P Z **B** **B** G
23 : G **B** P **B** Z Z P **B** *R* A P Q *R* Z **B** **B** G
24 : G **B** P **B** Z Z P Q G *R* P Z Q *R* **B** **B** G
25 : G **B** P **B** Z Z *R* A G A *R* Z Z **B** P **B** G
26 : G **B** P **B** Z *R* A *R* G *R* A *R* Z **B** P **B** G
27 : G **B** P **B** *R* A P **B** G **B** P A *R* **B** P **B** G
28 : G **B** P Q G *R* P **B** G **B** P *R* G Q P **B** G
29 : G **B** *R* A G A *R* **B** G **B** *R* A G A *R* **B** G
30 : G **B** G *R* G *R* G **B** G **B** G *R* G *R* G **B** G
31 : G G G G G G G G G G G G G G G G G
32 : F F F F F F F F F F F F F F F F F