Automatic Protoboard Layout

by

Michael Mekonnen

B.S. EECS, Massachusetts Institute of Technology (2013)
B.S. Mathematics, Massachusetts Institute of Technology (2013)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2014

Author .....
Department of Electrical Engineering and Computer Science
September 27, 2013

Certified by .....
Dennis M. Freeman
Professor
Thesis Supervisor

Certified by .....
Adam J. Hartz
Instructor
Thesis Supervisor

Accepted by .....
Professor Albert R. Meyer
Chairman, Department Committee on Graduate Theses

Automatic Protoboard Layout

by

Michael Mekonnen

Submitted to the Department of Electrical Engineering and Computer Science
on September 27, 2013, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Electrical Engineering and Computer Science

Abstract

Abstract.

Thesis Supervisor: Dennis M. Freeman
Title: *Professor*

Thesis Supervisor: Adam J. Hartz
Title: *Instructor*

Acknowledgments

Acknowledgments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

What problem are we solving?

1.2 Motivation

Why is this an interesting problem?

1.3 Goal

Precisely state the goal of this project. In particular, explain that we ultimately want to make a teaching tool for 6.01.

1.4 Outline

How is the Thesis organized?

Chapter 2

Background

2.1 Technical Background

What is a circuit schematic? What is a protoboard? What circuit components are we working with in this project?

2.2 Previous Work

2.2.1 Current tools in 6.01

Discuss CMax and its capabilities.

2.2.2 Current work in automatic protoboard layout

What similar work has been done before?

Chapter 3

Methods

In this Section, I discuss my solution to the problem and various alternatives I considered along the way.

3.1 Overview

I solved this problem by formulating it as a search problem. By this I mean, given a schematic of a circuit, I start from an empty protoboard, and I search through the space of all possible protoboard layouts to find the protoboard corresponding to the schematic at hand. The space of all possible protoboards is very large (?), so I utilize various simplifications and heuristics to facilitate the search.

I broke down the problem into two parts. The first task is finding a placement of all the circuit pieces on the protoboard. The second task is wiring them up appropriately.

3.2 Part 1: Piece Placement

Let us first consider how to place a set of circuit pieces on the protoboard for a given circuit schematic.

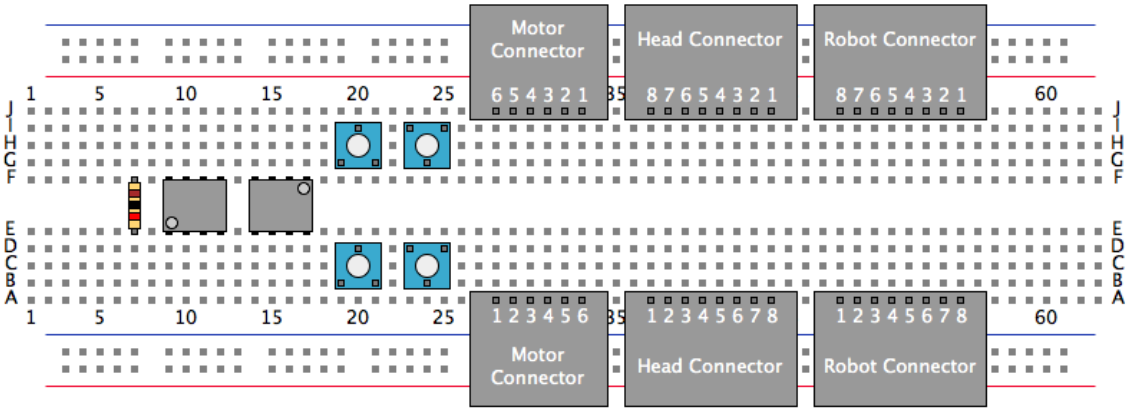

Figure 3-1: Various acceptable ways of putting each of the circuit pieces on the protoboard.

3.2.1 The Pieces

Any given circuit may contain resistors, Op Amps, pots, motors, head connector parts, or robot connector parts. For each of these components, we must put down a corresponding piece on the protoboard.

Resistors

For the sake of simplicity, and to significantly reduce the search space (?), for every resistor in the schematic, I use one resistor piece on the protoboard placed in the middle strip of the protoboard as shown in Figure 3-1. This choice, i.e. allowing the resistor pieces to only reside in the middle strip of the protoboard, is critical as the resistor pieces can generally be placed at numerous places on the protoboard. With this restriction, there are 63 slots available for one resistor. Without this restriction, there are a total of 763 slots available. The restirction is good when we consider the reduction in the search space size. On the other hand, the restriction is bad when we consider the size of circuits the algorithm can layout. Given that the number of resistors in a typical 6.01 circuit is very small (?), this restriction proves to be very useful, but we will consider the alternative in Section 3.4.

n	$f(n)$
1	1
2	3
3	7
4	25
5	81
6	331
7	1303
8	5937
9	26785
10	133651

Table 3.1: Number of ways of packaging together n Op Amps for various values of n.

Op Amps

Op Amps are the trickiest components to handle because each Op Amp package put on the protoboard contains two Op Amps within it. Thus, we face the task of packaging the Op Amps in the schematic in the "best" possible way, i.e. so as to require as little work as possible when wiring the pieces together. Equation 3.1 presents an expression for the value $f(n)$, the number of different ways to package together n Op Amps. For example, if we have 2 Op Amps, we can either use one Op Amp package for each, or put them both in the same package, which we can do in one of two different ways. Hence, $f(2) = 3$. To get a sense of how many different packagings are possible, Table 3.2.1 gives the values of $f(n)$ for various n.

$$f(n) = \sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} \frac{n!}{k!(n-2k)!} \tag{3.1}$$

Each Op Amp package is placed in the middle strip of the protoboard, with two acceptable orientations, as shown in Figure 3-1.

Pots

Each pot piece can be placed in one of two vertical locations on the protoboard. Each pot piece also has two possible orientations. Figure 3-1 provides examples of all acceptable options.

Head, Motor, and Robot Connectors

We use a 6-pin connector to connect to a motor, and an 8-pin connector to connect to either a head or a robot. Each connector can be placed in one of two vertical locations on the protoboard, as shown in Figure 3-1.

3.2.2 Choosing a Placement

When choosing a placement of circuit pieces on the protoboard, we have at hand a plethora of options. First we must choose among a possibly large number of ways to package together the Op Amps in the circuit. For each possible packaging of Op Amps, we must consider various ways of placing the pieces on the protoboard.

Simplifications

I reduce this large number of options by only allowing placements in which no two pieces share a column. Once again, this is not necessary in general, but the number of pieces necessary for a typical 6.01 circuit would certainly fit in this framework.

Next, I specify that there be exactly two columns on the protoboard separating each consecuitive pair of pieces, unless the pieces are both resistors, in which case there must be exactly one column separating them. These numbers of columns were chosen to leave enough space for wiring. Given a set of pieces to be put on the protoboard, this specification reduces the problem of choosing a placement for the pieces to finding an *order* of the pieces together with choosing their respective vertical locations and orientations.

Given these simplifications, we have various options as to how to pick a placement.

Random Placement

One simple alternative may be to choose a placement randomly. That is, we choose an Op Amp packaging randomly; we choose an order of the pieces randomly; and we choose the vertical locations and orientations of the pieces randomly as well. The advantage of this approach is that it gives us a placement very quickly without

requiring much computation. On the other hand, we may end up placing two pieces that need to be connected to each other very far apart, and we will have a difficult time doing the wiring. Hence, we ought to consider alternatives in which we take into account the task of wiring. We should try to place the pieces so as to require as little work during wiring as possible.

Minimal Heuristic Cost

The key idea is that if two pieces are meant to be connected together by wires, then they ought to be placed close to each other on the protoboard. We can capture this idea by assigning heuristic costs to the placements and choosing a placement that produces the minimal heuristic cost.

Let us first devise the cost function to achieve this goal. Given a circuit schematic and a corresponding placement of the circuit pieces on the protoboard, what do we need to connect with wires? Well, every pair of components in the schematic that are connected by a wire gives us a corresponding pair of locations on the protoboard that ought to be connected by wires. However, we can express this requirement a little bit more concisely. We ought to consider all of the nodes in the schematic, and find the circuit components in the schematic that are connected to the respective nodes. Now for each node in the circuit, we get a set of locations on the protoboard that ought to be interconnected. The first step in devising the cost function we are looking for is to have a way to estimate the cost of connecting two locations on the protoboard. A simple such cost function that comes to mind is the Manhattan distance between the two locations. Recall that we want to produce aesthetically pleasing protoboard layouts, and one of the requirements in achieving this goal is only using horizontal and vertical wires (i.e. no diagonal wires) so the Manhattan distance cost is appropriate. Given this heuristic cost for connecting two locations with wires, we can define the heuristic cost for interconnecting the locations associated with a particular node to be the weight of the minimum spanning tree of the locations. Now we can define the cost of a placement to be the sum over all nodes in the circuit of the cost for interconnecting the locations for each node.

Now that we have a cost function for placements, we can aim to find a placement with the minimal cost. However, this involves trying all possible orderings of the pieces with which we are working. For example, if we are trying to order 10 pieces, we would need to look at $10! = 3628800$ possible orderings (?). Note that this is in addition to searching over all possible ways of packaging the Op Amps together. It is clear to see that the search for a minimal cost placement quickly gets out of hand. So we aim to find a placement that has a very small, though maybe not minimal, cost.

Small Heuristic Cost

Algorithm 1 presents a polynomial-time procedure that orders a given list of pieces in a way that results in a small cost. The algorithm places one of the pieces at a time, starting from an empty placement. It relies on two ideas. First, once a piece has been placed, all the pieces that are connected to it will be placed soon after so that it is more likely that those pieces are placed close to it. Second, we place the pieces with the most nodes first since those are the ones that most likely have connections with many other pieces.

Algorithm 1: Producing a circuit piece placement with small heuristic cost.

Data: A list P of circuit pieces.
Result: A list R of circuit pieces representing a placement.

```
Sort P in decreasing number of nodes on the respective pieces
Q ← empty Queue
R ← empty List
while P is not empty do
    Pop the first piece in P and push it onto Q
    while Q is not empty do
        p ← Q.pop()
        Consider all vertical locations and orientations of p
        Place p at an index in R that minimizes the cost of P
        foreach piece q in P connected to p do
            Pop q out of P and push it onto Q
```

Using one of the above methods, we can find a placement of circuit pieces on a protoboard. Our next task is wiring them together to produce a circuit equivalent to

the circuit schematic of interest.

3.3 Part 2: Wiring

In the previous section we discussed what locations we need to wire together: for every node in the circuit, we get a set of locations on the protoboard that need to be interconnected. The question now is how to achieve this wiring. We approach the problem as a search problem and use the $A*$ search algorithm to solve it.

3.3.1 Using $A*$

When using the $A*$ algorithm, we need to design four things:

1. The notion of a vertex[1] in the search tree, the cost associated with a vertex, and how we obtain the neighbors of a vertex,
2. The starting vertex,
3. How we identify whether a particular vertex in the search tree achieves the goal of the search, and
4. A heuristic function that estimates the distance from a given vertex to a goal vertex.

3.3.2 Vertices

Each vertex will hold a representation of some protoboard. Each representation will contain all of the pieces, and possibly a set of wires interconnecting the pieces. The starting vertex will have all of the pieces but no wires.

We obtain the neighbors of a vertex by taking the current protoboard and producing new ones in which we place exactly one wire at various locations. We choose the starting point of a wire to be any one of the free locations on the protoboard

[1]The prefered name is "node" but I will use vertex since we already use node to refer to nodes in circuits.

that is already connected to one of the pieces, and we extend the wires in all possible vertical and horizontal directions up to some fixed wire length. Note that we need to take great care when placing wires in order not to short different nodes. We discard any vertices that arise from placing a wire that shorts two different nodes.

The way we define the cost of a vertex, i.e. the cost of getting from the starting vertex to a vertex of interest, depends on what we consider to be an aesthetically pleasing protoboard layout. In general, we want to penalize having long wires, many wires, or crossing wires. In my implementation, while I have a large penalty for two crossing wires of opposite orientations (i.e. vertical and horizontal), I do not allow crossing wires that have the same orientation as this configuration is particularly difficult to physically build and debug. Finally, we want to favor making a desired interconnection between locations on the protoboard. I chose my penalties experimentally (?).

Each vertex will not only hold a protoboard, but it will also hold a set of pairs of locations on the protoboard that need to be connected by wires. Each pair (loc_1, loc_2) of locations tells us that we need to have a set of wires connecting some location connected to loc_1 to some location connected to loc_2.

An important consideration we need to make is how we want to organize the search. Recall that we have a set of nodes in the circuit of interest, and for each node we have a set of locations that need to be interconnected. Given this information, we may choose one of the following three strategies to carryout the search:

1. For each node, collect a set of pairs of locations on the protoboard corresponding to a minimum spanning tree of the locations for that node, so that if all pairs of locations in this spanning tree are connected, then the locations for the node will be interconnected. Collect all such pairs of locations for all of the nodes in the circuit, and have the starting vertex hold this set of pairs of locations.

2. Treat each node separately. That is, iteratively interconnect the locations for each of the nodes until there are no more nodes in the circuit.

3. Treat each pair of locations that needs to be connected separately. That is,

iteratively connect pairs of locations that need to be connected until there are no more pairs.

The choice of one of these strategies has a significant effect on the outcome of the search. We will discuss the difference in detail in Chapter 4.

3.3.3 Goal test

Given a vertex, we know that it is a goal vertex if all of the pairs of locations it holds are already connected in the protoboard it holds.

3.3.4 Search heuristic

In $A*$ search, choosing the right heuristic can often make the search much more efficient. In our problem, one option we have is not to use a heuristic, and that alternative will be explored in Chapter 4. However there is a natural heuristic that suggests itself that we ought to consider. Given a vertex, we can estimate its distance from a goal as follows. For each pair of locations (loc_1, loc_2) that need to be connected, we could consider its distance from the goal to be the smallest Manhattan distance between any location connected to loc_1 and any other location connected to loc_2. To compute the heuristic cost of a vertex, we simply add up this value for each of the pairs of locations that need to be connected. Chapter 4 presents the performance of this heuristic verses using no heuristic.

3.4 Treating Resistors as Wires

The discussion in Section 3.2.1 presented that we treat resistors just as we do the other components. That is, we give each resistor a fixed place on the protoboard in the first step of the algorithm before the wiring step. However, resistors have the special property among the circuit pieces that they can be thought of as wires of length 3. Hence, it may be possible to handle resistors in the wiring step instead of the placement step. Chapter 4 presents a comparison of these two approaches.

Note that treating resistors in the wiring step is not a trivial task. First, there may be nodes in the circuit that are connected to some resistors, but no other pieces. In this case, we must be sure to reserve space on the protoboard for that node as the wiring step relies on the presence of each node on the protoboard. Second, we must keep careful track of pairs of locations that need to be connected by simple wires and pairs of locations that need to be connected using resistors.

3.5 Evaluation Method

Here we present how we go about evaluating a particular solution to the problem. How can we tell if a layout tool is good? In particular, how can we tell if a layout tool is good enough for the purposes of 6.01 labs. To answer these questions well, we need to test the layout tool on numerous schematics and analyze its performance on laying out those schematics. As manually generating numerous test schematics is tedious and very time-consuming, we devised a method to randomly generate thousands of test schematics.

The random schematic generation goes as follows. We create 6 basic sub-parts of schematics. These 6 bases are depicted in Figure 3-2. These bases cover all of the components that may be necessary in a 6.01 circuit. Each sub-part also offers at least 3 points of connection with other subparts. The random generation algorithm takes all possible combinations of 6 bases, allowing for repetition of bases. The only bases that can appear at most once in a schematic are the Head Connector and Robot Connector - there is no need for more that one of each of these in 6.01 labs. All other bases may appear up to 6 times. For a given combination of bases, we generate 10 schematics as follows. For each number of connections k from 0 to 9, we generate a schematic in which we put k randomly chosen wires interconnecting the bases in the combination.

This scheme produces a total of 5242 test schematics out of a possible total of 889785038484423775.

Figure 3-2: TODO

Chapter 4

Results

In this chapter, we compare the various alternatives discussed in Chapter 3. We test each of the alternatives on the randomly generated dataset containing (?)test schematics. All of the alternatives are evaluated on this same dataset for appropriate comparison.

To assess the goodness of an alternative method, we give:

1. A diagram of a 2D array of points in which each point represents a run. Each point is colored white or red corresponding to a successful or unsuccessful run, respectively. This array gives a general overview of how well the method performed on the test dataset.

2. The success rate of the method on the test dataset. The higher this value, the better the method.

3. The average and standard deviation of success times. The lower the average, the better the method.

4. The average and standard deviation of failure times. The lower the average, the better the method.

5. Among the solved schematics, the average and standard deviation of the number of wires used. The lower the average, the better the method.

6. Among the solved schematics, the average and standard deviation of the total wire length of the wires on the protoboard. The lower the average, the better the method.

7. Among the solved schematics, the average and standard deviation of the number of wire crosses on the protoboard. The lower the average, the better the method.

To assess the effect of various aspects of schematics on the method, we give tables in which we consider the values of each of the following items, and record how success rate, average success time, and average failure time depend on the values:

1. Number of components.

2. Number of nodes in the circuit.

3. Number of resistors.

4. Number of Op Amps.

5. Number of pots.

6. Number of motors.

7. Number of Head Connectors (0 or 1).

8. Number of Robot Connectors (0 or 1).

Analysis of the results follows in Chapter 5.

4.1 Resistors as Components

First we consider the case in which we treat resistors as components. That is, the resistors in the circuit will be placed on the protoboard by the placement step. With this setup, we consider 5 different ways of wiring.

4.1.1 All connections at once

Here we consider connecting all pairs of location that need to be connected at once. Figure 4-1 displays the overall results. Table 4.1.1 presents the statistics we obtain with this setup.

Figure 4-1: Overall results when we treat resistors as components and connect all pairs of protoboard locations at once.

Item	Value
Success rate	
Solve time mean and standard deviation	
Failure time mean and standard deviation	
Number of wires mean and standard deviation	
Total wire length mean and standard deviation	
Total wire crosses mean and standard deviation	

Table 4.1: Test statistics when we treat resistors as components and connect all pairs of protoboard locations at once.

4.1.2 Connections for one node at a time, node with *most* connections first

Next we consider connecting the location pairs for one node at a time, ordered in *descending* order of number of location pairs per node. Figure 4-2 shows the overall results, and Table 4.1.2 presents the statistics for this setup.

Figure 4-2: Overall results when we treat resistors as components and connect the location pairs for one node at a time, with the node with the most connections going first.

Item	Value
Success rate	
Solve time mean and standard deviation	
Failure time mean and standard deviation	
Number of wires mean and standard deviation	
Total wire length mean and standard deviation	
Total wire crosses mean and standard deviation	

Table 4.2: Test statistics when we treat resistors as components and connect the location pairs one node at a time, with the node with the most connections going first.

4.1.3 Connections for one node at a time, node with *least* connections first

Here also, we consider connecting the location pairs for one node at a time, but this time ordered in *ascending* order of number of location pairs per node. Figure 4-3 shows the overall results, and Table 4.1.3 presents the statistics for this setup.

Figure 4-3: Overall results when we treat resistors as components and connect the location pairs for one node at a time, with the node with the least connections going first.

Item	Value
Success rate	
Solve time mean and standard deviation	
Failure time mean and standard deviation	
Number of wires mean and standard deviation	
Total wire length mean and standard deviation	
Total wire crosses mean and standard deviation	

Table 4.3: Test statistics when we treat resistors as components and connect the location pairs one node at a time, with the node with the least connections going first.

4.1.4 One connection at a time, *shortest* connection first

Now we look at wiring one pair of locations at a time. First, we consider starting from the closest pair of locations. Figure 4-4 shows the overall results and Table 4.1.4 presents the statistics for this setup.

Figure 4-4: Overall results when we treat resistors as components and connect one pair of locations at a time, starting with the closest pair of locations.

Item	Value
Success rate	
Solve time mean and standard deviation	
Failure time mean and standard deviation	
Number of wires mean and standard deviation	
Total wire length mean and standard deviation	
Total wire crosses mean and standard deviation	

Table 4.4: Test statistics when we treat resistors as components and connect one pair of locations at a time, starting with the closest pair of locations.

4.1.5 One connection at a time, *longest* connection first

Here we connect one pair of locations a time, but this time we start with the farthest pair of locations. Figure 4-5 shows the overall results and Table 4.1.5 presents the statistics for this setup.

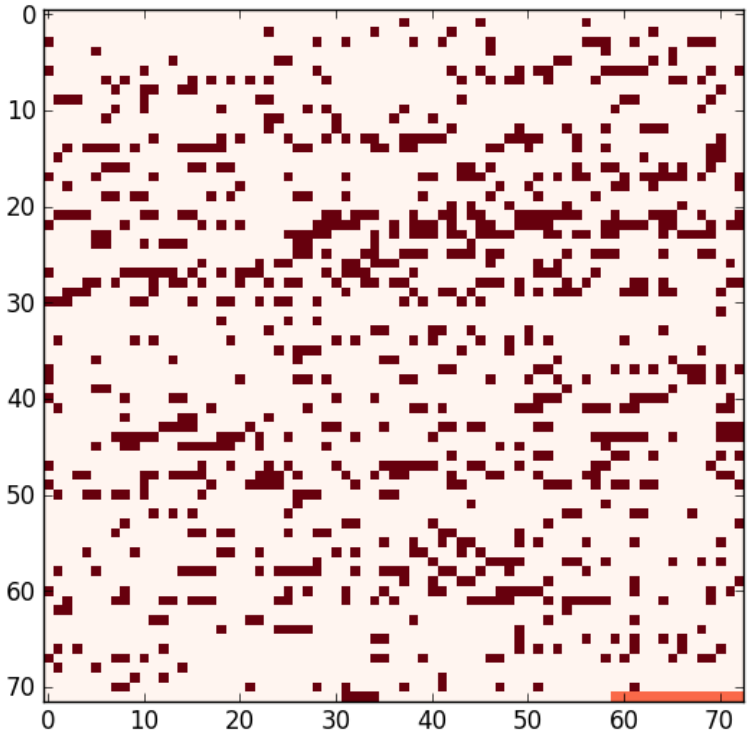

Figure 4-5: Overall results when we treat resistors as components and connect one pair of locations at a time, starting with the farthest pair of locations.

Item	Value
Success rate	
Solve time mean and standard deviation	
Failure time mean and standard deviation	
Number of wires mean and standard deviation	
Total wire length mean and standard deviation	
Total wire crosses mean and standard deviation	

Table 4.5: Test statistics when we treat resistors as components and connect one pair of locations at a time, starting with the farthest pair of locations.

4.1.6 Choice

With resistors being treated as components, the test results suggest that the best way to wire the components is (?). Next we look at the effects of various aspects of the schematics on this method.

Nodes	Sample size	Success rate	Success time mean	Failure time mean
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

Table 4.6: TODO

Components	Sample size	Success rate	Success time mean	Failure time mean
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

Table 4.7: TODO

Resistors	Sample size	Success rate	Success time mean	Failure time mean
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

Table 4.8: TODO

Op Amps	Sample size	Success rate	Success time mean	Failure time mean
0				
1				
2				
3				
4				
5				
6				

Table 4.9: TODO

Pots	Sample size	Success rate	Success time mean	Failure time mean
0				
1				
2				
3				
4				
5				
6				

Table 4.10: TODO

Motors	Sample size	Success rate	Success time mean	Failure time mean
0				
1				
2				
3				
4				
5				
6				

Table 4.11: TODO

Head Connectors	Sample size	Success rate	Success time mean	Failure time mean
0				
1				

Table 4.12: TODO

Robot Connectors	Sample size	Success rate	Success time mean	Failure time mean
0				
1				

Table 4.13: TODO

4.2 Resistors as Wires

Now we consider the case in which we treat resistors as wires. That is, the resistors in the circuit will not be placed on the protoboard by the placement step. Rather, we will place them in the wiring process as described in Section 3.4. With this setup, we consider the 5 different ways of wiring once more.

4.2.1 All connections at once

Figure 4-6 displays the overall results and Table 4.2.1 presents the statistics we obtain with this setup.

Figure 4-6: Overall results when we treat resistors as wires and connect all pairs of protoboard locations at once.

Item	Value
Success rate	
Solve time mean and standard deviation	
Failure time mean and standard deviation	
Number of wires mean and standard deviation	
Total wire length mean and standard deviation	
Total wire crosses mean and standard deviation	

Table 4.14: Test statistics when we treat resistors as wires and connect all pairs of protoboard locations at once.

4.2.2 Connections for one node at a time, node with *most* connections first

Figure 4-7 shows the overall results, and Table 4.2.2 presents the statistics for this setup.

Figure 4-7: Overall results when we treat resistors as wires and connect the location pairs for one node at a time, with the node with the most connections going first.

Item	Value
Success rate	
Solve time mean and standard deviation	
Failure time mean and standard deviation	
Number of wires mean and standard deviation	
Total wire length mean and standard deviation	
Total wire crosses mean and standard deviation	

Table 4.15: Test statistics when we treat resistors as wires and connect the location pairs one node at a time, with the node with the most connections going first.

4.2.3 Connections for one node at a time, node with *least* connections first

Figure 4-8 shows the overall results, and Table 4.2.3 presents the statistics for this setup.

Figure 4-8: Overall results when we treat resistors as wires and connect the location pairs for one node at a time, with the node with the least connections going first.

Item	Value
Success rate	
Solve time mean and standard deviation	
Failure time mean and standard deviation	
Number of wires mean and standard deviation	
Total wire length mean and standard deviation	
Total wire crosses mean and standard deviation	

Table 4.16: Test statistics when we treat resistors as wires and connect the location pairs one node at a time, with the node with the least connections going first.

4.2.4 One connection at a time, *shortest* connection first

Figure 4-9 shows the overall results and Table 4.2.4 presents the statistics for this setup.

Figure 4-9: Overall results when we treat resistors as wires and connect one pair of locations at a time, starting with the closest pair of locations.

Item	Value
Success rate	
Solve time mean and standard deviation	
Failure time mean and standard deviation	
Number of wires mean and standard deviation	
Total wire length mean and standard deviation	
Total wire crosses mean and standard deviation	

Table 4.17: Test statistics when we treat resistors as wires and connect one pair of locations at a time, starting with the closest pair of locations.

4.2.5 One connection at a time, *longest* connection first

Figure 4-10 shows the overall results and Table 4.2.5 presents the statistics for this setup.

Figure 4-10: Overall results when we treat resistors as wires and connect one pair of locations at a time, starting with the farthest pair of locations.

Item	Value
Success rate	
Solve time mean and standard deviation	
Failure time mean and standard deviation	
Number of wires mean and standard deviation	
Total wire length mean and standard deviation	
Total wire crosses mean and standard deviation	

Table 4.18: Test statistics when we treat resistors as wires and connect one pair of locations at a time, starting with the farthest pair of locations.

4.2.6 Choice

With resistors being treated as wires, the test results suggest that the best way to wire the components is (?). Next we look at the effects of various aspects of the schematics on this method.

Nodes	Sample size	Success rate	Success time mean	Failure time mean
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

Table 4.19: TODO

Components	Sample size	Success rate	Success time mean	Failure time mean
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

Table 4.20: TODO

Resistors	Sample size	Success rate	Success time mean	Failure time mean
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

Table 4.21: TODO

Op Amps	Sample size	Success rate	Success time mean	Failure time mean
0				
1				
2				
3				
4				
5				
6				

Table 4.22: TODO

Pots	Sample size	Success rate	Success time mean	Failure time mean
0				
1				
2				
3				
4				
5				
6				

Table 4.23: TODO

Motors	Sample size	Success rate	Success time mean	Failure time mean
0				
1				
2				
3				
4				
5				
6				

Table 4.24: TODO

Head Connectors	Sample size	Success rate	Success time mean	Failure time mean
0				
1				

Table 4.25: TODO

Robot Connectors	Sample size	Success rate	Success time mean	Failure time mean
0				
1				

Table 4.26: TODO

4.3 Best First Search

Here we compare using $A*$ search to using Best First Search. We look at the best wiring strategies we have found when treating resistors as components and when treating resistors as wires, and for each we will look at the results of using Best First Search instead of $A*$.

4.3.1 Resistors as Components Winner

First we will employ Best First Search while treating resistors as components. Recall that we found the best wiring strategy in this setup to be (?), hence we will use that strategy here, but with $A*$ replaced by Best First Search. Figure 4-11 shows the overall results, and Table 4.3.1 presents the statistics for this setup.

Figure 4-11: Overall results when we treat resistors as components and (?)using Best First Search.

Item	Value
Success rate	
Solve time mean and standard deviation	
Failure time mean and standard deviation	
Number of wires mean and standard deviation	
Total wire length mean and standard deviation	
Total wire crosses mean and standard deviation	

Table 4.27: Test statistics when we treat resistors as components and (?)using Best First Search.

4.3.2 Resistors as Wires Winner

Next we will employ Best First Search while treating resistors as wires. Recall that we found the best wiring strategy in this setup to be (?), hence we will use that strategy here, but, once again, with $A*$ replaced by Best First Search. Figure 4-12 shows the overall results, and Table 4.3.2 presents the statistics for this setup.

Figure 4-12: Overall results when we treat resistors as wires and (?)using Best First Search.

Item	Value
Success rate	
Solve time mean and standard deviation	
Failure time mean and standard deviation	
Number of wires mean and standard deviation	
Total wire length mean and standard deviation	
Total wire crosses mean and standard deviation	

Table 4.28: Test statistics when we treat resistors as wires and (?)using Best First Search.

Chapter 5

Discussion

5.1 Explaining the Results

Give plausible explanation for the observed results.

5.2 Remarks

Why are these results encouraging? What are their implications? Relate back to Introduction to Thesis. What could have been done differently?

Appendix A

Schematic Drawing GUI

Discuss the features and capabilities of the GUI.