Learning the Structure of Local Neural Circuits in Mouse Ectorhinal Cortex

Scott Brookes and Derek Racine

May 19, 2014

1 Introduction

The brain is structured such that local networks of neurons can perform distinct, modular, tasks. For example, the ectorhinal cortex is known to receive the majority of its input from visual sensory areas and is involved in visual memory and object recognition. However, the network structure is not well understood. Training an animal to recognize certain images, such as through visual paired-associate learning, might also affect the network dynamics. In particular, one might hypothesize that this training could cause neurons to become tuned to images that the mouse has seen before, but not others. That is, the activity of neurons would be well correlated with stimuli used during training. Yet, this relationship hasn't been explored.

Simplified neurons can be described as being in one of two states: resting or firing. When a neuron fires, calcium ions flow into it from the extracellular fluid. As such, fluorescence observed from calcium sensors can serve as a proxy for neural activity.

The recent discovery of an ultrasensitive family of fluorescent calcium sensors, GCaMP6, has caused an explosion in the availability of time-series data based on the use of this technique.[1] Whereas previous recording methods were limited to only a few cells, fluorescent imaging techniques can measure the activity of an entire region of neurons simultaneously, as shown in Figure 1. This data offers an opportunity to learn the structure of local neural circuits through statistical methods. Indeed, previous work has addressed this problem by modeling fluorescent imaging data as a collection of coupled Hidden Markov chains, one for each neuron.[2]

2 Dataset

2.1 Description

We have obtained data sets from the Max Planck Institute for Neurobiology corresponding to three recording sessions from the same mouse. Each recording session consists of multiple time series of fluorescent activity corresponding to the neurons within the imaged area. GCaMP6s, a slower decaying member of the GCaMP6 family, was used as the calcium sensor in these experiments. The first data set consists of activity of neurons in the primary visual cortex (V1) that are known to reliably respond to oriented gratings at a certain angle (see Figure 2b), which are presented to the mouse during the trial. An illustration of this setup is shown in Figure 2a. This data will be a useful control for testing our structure learning method because we expect that there will be a strong correlation between neural activity and the display of specific stimuli.

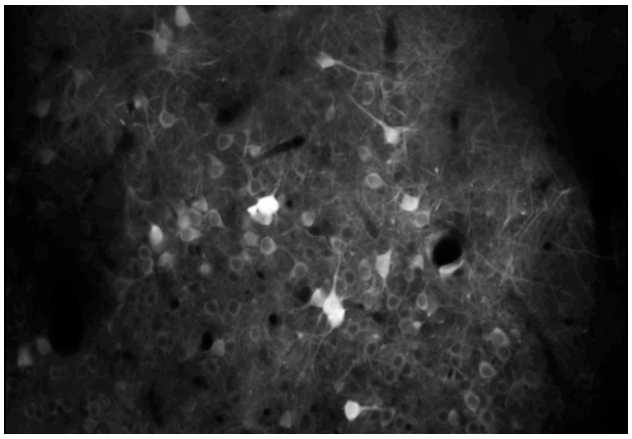
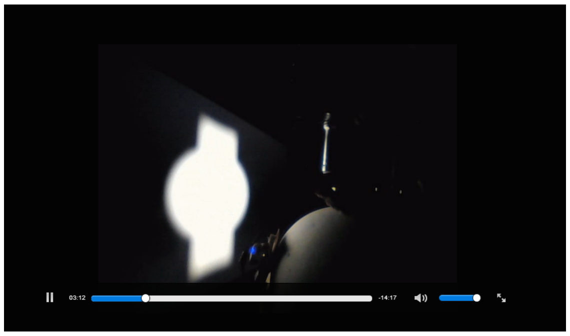
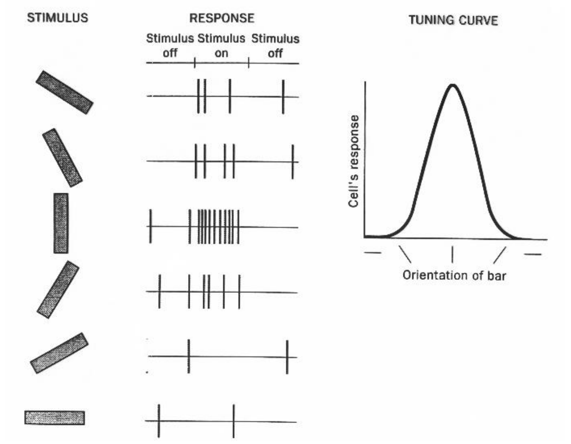


Figure 1: A snapshot of neurons labeled with GCaMP6s within a region of the cortex.



(a) Stimuli being Presented to the Mouse



(b) Typical Response to Moving Gratings Stimuli

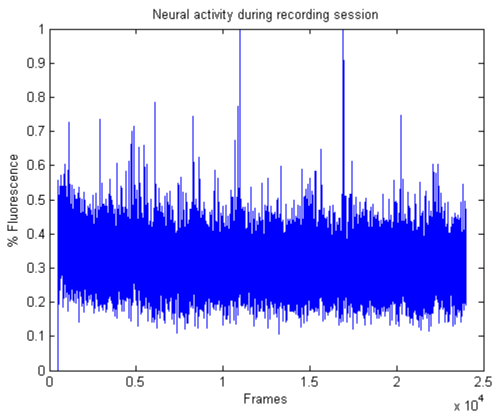
Figure 2: Showing the Mouse Stimuli and Measuring its Response

The remaining data sets track the activity of neurons in the ectorhinal cortex during periods of spontaneous activity and the display of novel stimuli. Inferring network structure for these time series will allow us to investigate the difference between the correlation neurons exhibit during spontaneous activity versus when a stimulus is present.

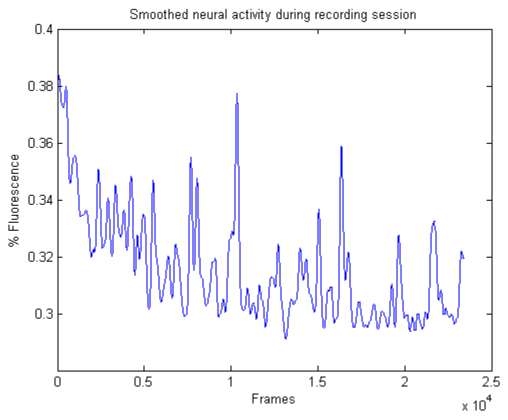
2.2 Pre-Processing

3 Method

We modelled the local neural circuits as a Bayesian network and infered the edges from the correlations of the activity of individual neurons, which we treated as binary random variables – firing or not firing. In order to learn the structure, we discretized our continuous series into buckets that each represent a single instance of the overall network.



(a) Raw Neural Activity over Time



(b) Smoothed Neural Activity over Time

Figure 3: Smoothing Neural Activity

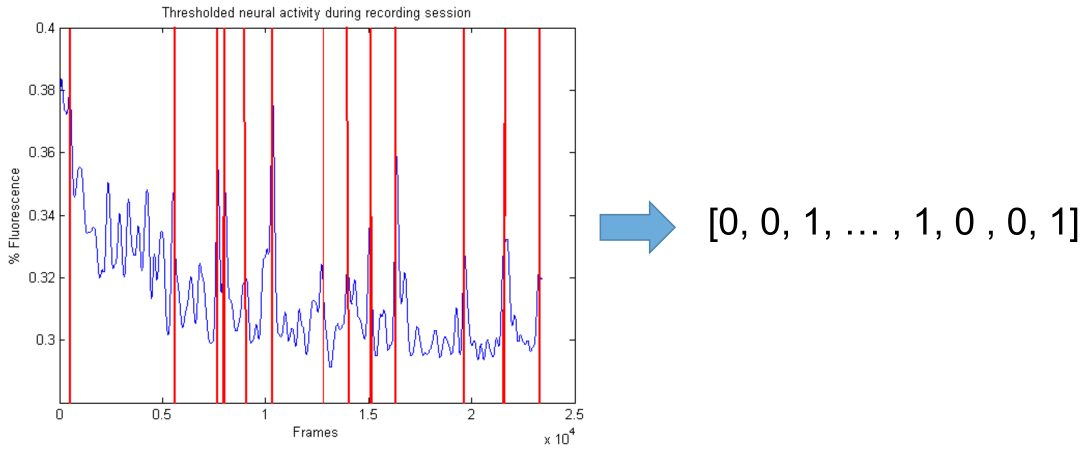


Figure 4: Detecting Spikes in the Neural Activity

We are using the GlobalMIT toolbox for learning the globaly optimal Dynamic Bayes Net structure.[3] The toolbox uses the recently introduced *mutual information test (MIT)*. It is implemented in both Matlab® and C++.

3.1 The Mutual Information Test

We will present just a brief overview of this method. For a full discussion of this algorithm, we direct an interested reader to.[4]

A potential network is scored by the total mutual information shared between each node and its parents, less a term that quatifies how statistically significant the shared information is. The full discussion shows that the first term, considering the total mutual information shared between each node and its parents, is equivalent to maximizing the log-likelihood criterion. As we know, however, learning a Bayes Net using maximum log-likelihood alone risks overfitting the data. In order to avoid this unnecessary complexity, the MIT method considers whether the information gained by adding a parent to a node is statistically significant. This effectively penalizes a complex model.

3.2 Using the Toolkit

Using the toolkit is rather simple. In order to illustrate the power of the toolkit, we give two toy examples that illustrate the output of the toolkit given some input. For the output shown in Figure 5a, the input is a data matrix shown in Table 1 where that table has different timeslices as columns and different variables as rows. The same format is used to display the data table shown in Table 2 used for the toy 2 example shown in Figure 5b.

1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Table 1: Toy1 Dataset

1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1

Table 2: Toy2 Dataset

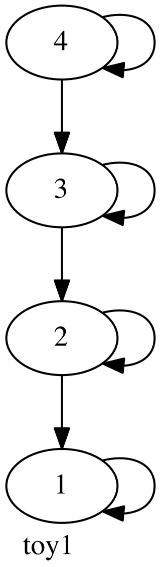
3.3 Implementation Considerations

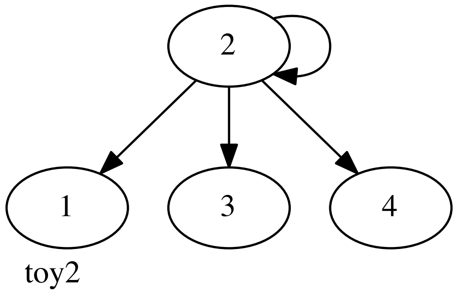
The toolkit takes a parameter $\alpha : 0 < \alpha \leq 1$ that tunes the rigor to which the algorithm will try to fit the model. Lower α's allow for a lower degree of uncertainty, so a lower α generates a more accurate result but also requires a great deal more computation time. We wanted to use as low an α as possible, but due to the data sizes found that the lowest we could realistically go was 0.99. This actually turns out to be pretty good; the α values used to illustrate how to use the model in the toolkit's documentation are 0.95, 0.999, and 0.9999. This leaves us feeling satisfied with the α we were able to use.

In order to model the entire neural network that we are interested in, we needed to make provisions for minimizing the complexity. The paper[3] demonstrates a problem in which there are 20 variables and 2,000 observations. In the Matlab® version of the toolkit this model takes more than a full day to be learned. Our dataset, naively, has as many as 100 variables with 24,000 observations.

In order to achieve practical computation times, we had to preform data preprocessing as discussed in section 2.2 of this paper including bucketing to get the number of observations down. In addition, the paper discusses that the C++ implementation of the toolkit takes only an hour to process this same example, leading us to compile and make use of the C++ implementation. This was more challenging than using just the tools on the Matlab® side.

Unfortunately, the toolkit made our initial progress very slow. We found that with most of permutations of data that we passed to the toolkit, it would crash due to memory errors. We compiled the C++ routine and were calling it from Matlab® as per the suggestion in the toolkit's documentation, but runs were still taking over an hour and could simply die an hour into the computation. In order to solve this problem, we used GDB to determine where the crash was coming from.





(b) Toy2 Example Output

(a) Toy1 Example Output

Figure 5: Toy Example Outputs

```
int Pstar=findPstar(g_score,2*Ne*HX_arr[i]);
int* powo= new int[Pstar-1];
```

(a) Bug in the toolbox

```
int Pstar=findPstar(g_score,2*Ne*HX_arr[i]);
if ( Pstar < 1 )
  continue; /* move to next node */
int* powo= new int[Pstar-1];
```

(b) Bug fix

The bug, shown in Figure 6a, was easy to fix (once found). The routine calculates P^*. Speaking loosely, this is the max number of parents that the algorithm is willing to assign to a node. Then, it allocates an array of size $P^* - 1$. If $P^* == 0$, there you have your memory error. The fix was very simple, as shown in Figure 6b. Since $P^* == 0$ implies that the node has no possible parents, we can simply skip the computation on that node and avoid the problematic allocation.

Although a trivial fix, finding this bug added a lot of time to our work. Once we found it, despite having already presented our project to the class, we were able to get some really interesting results as you can see in this paper.

4 Results

We expected that the structure we learned from the first data set will reflect the high correlation between neural activity and the presence of particular stimuli in V1. After we verified that this was indeed the case, we applied the same method to the other data sets in order to learn the network structure of the neural circuit in the ectorhinal cortex. A detailed quantitative description of our results follows:

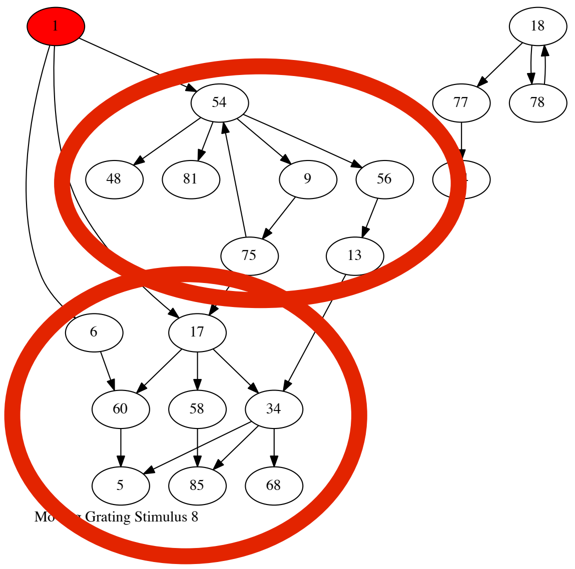


Figure 7: Clustering Revealed in the Primary Visual (V1) Cortex

Moving Grating Stimulus Clustering Coefficient	1 0.0079	2 0	3 0.0109	4 0.0099	5 0	6 0	7 0	8 0	
	9 0.0022	10 0.0139	11 0.0115	12 0.0047	13 0.0118	14 0	15 0.0081	16 0	Average 0.0051

Table 3: Clustering Coefficients for the Primary Visual Cortex with Moving Grating Stimuli

Stimulus	1	2	3	4	5	6	7	8		
CC with	0.0057	0.0203	0.0322	0.0205	0.0062	0.0195	0.0233	0		
CC without	0.0215	0.0174	0.0124	0.0104	0.0027	0.011	0.0097	0.0124		
	9	10	11	12	13	14	15	16	Average	Varience
	0.01	0.0118	0.0143	0.01	0.0158	0	0.0033	0.0042	0.0123	7.85×10^{-5}
	0.0226	0.019	0.0384	0.037	0.0136	0.0332	0.0323	0.0163	0.0194	0.0001

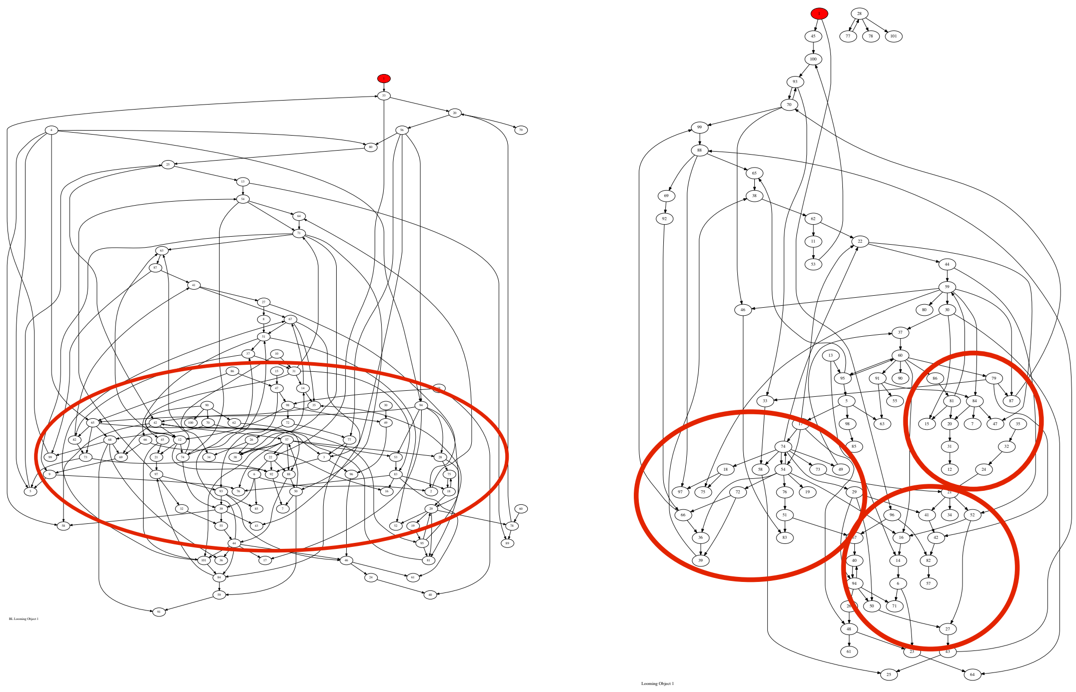
Table 4: Clustering Coefficients (CC) for the Ectorhinal Cortex with and without Looming Object Stimuli

5 Future Work

With more time, we would suggest a few different ways to take this work even further.

First, we could tune the hyperparameter α. This parameter encodes the degree to which the algorithm is going to penalize complexity. We chose a parameter based primarily on runtime. It is possible that a more closely tuned α could lead to better results.

Additionally, we would have liked to get our hands on a dataset from a third recording session in the ectorhinal cortex. The two that we processed were spontaneous activity and during the presentation of novel stimuli. We think that it would be interesting to examine the response of the same neurons during the



(a) Spontaneous Activity — No Clustering (b) Looming Object — Clear Clustering

Figure 8: Clustering in the Ectorhinal Cortex

presentation of learned stimuli. As explained in our introduction, we could hypothesize that the neurons would become tuned as stimuli are learned. We think that our work could be the starting point for exploring that hypothesis.

In order to make our problem tractable, we made some simplifying assumptions. For example, we treated the neurons as firing or not firing. In reality, the connections between neurons is not binary. In the future, we think that trying to infer edge weights in addition to network structure could provide even more valuable insight into the connectivity of these neural networks.

Finally, we propose work completely different from what we've done. We think that it could be interesting to use a hidden markov model to attempt to infer the presence of a specific stimulus given only the neural response data from the mouse. Although applications seem far-fetched now, we can imagine a world in which using just the activity in the brain to reproduce what the eyes are seeing could be very useful. Even know, it would be very cool.

6 Conclusion

Overall, we are pleased with the work that we did on this project. We were able to use the power of probabalistic graphical models to draw very interesting conclusions about the structure of local neural circuits in mouse ectorhinal cortex. We consider this project a success; we found interesting results while solidifying our understanding of topics that we studied in the course. Although there is a lot of room for future work, we think that our progress in this area was somewhat novel and a good start towards understanding neural

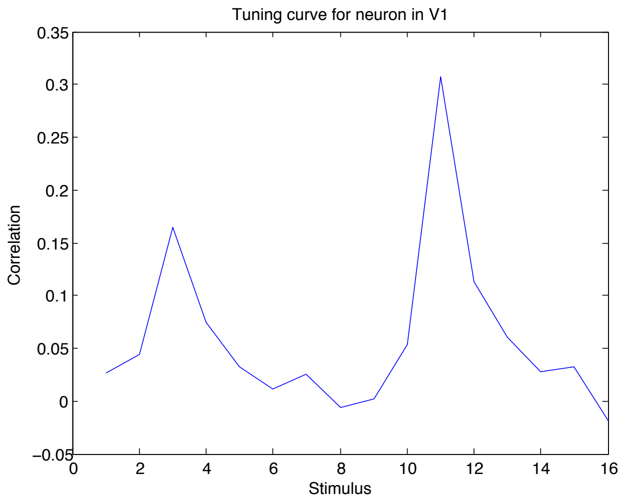


Figure 9: Our Measured Tuning Curve for the Moving Gratings

circuits in mouse ectorhinal cortex.

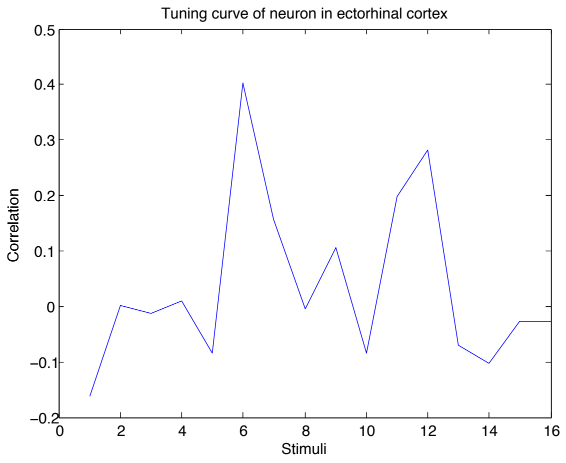
References

[1] Tsai-Wen Chen and et al. Ultrasensitive fluorescent proteins for imaging neuronal activity. *Nature*, 499.7458:295–300, 2013.

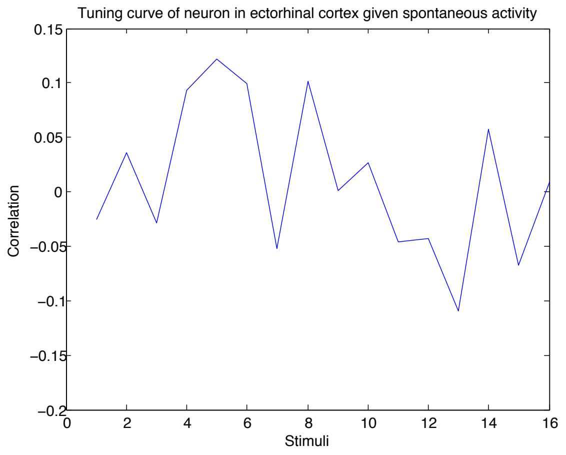
[2] Yuriy Mishchenko, Joshua T. Vogelstein, and Liam Paninski. A bayesian approach for inferring neuronal connectivity from calcium fluorescent imaging data. *The Annals of Applied Statistics*, 5.2B:1229–1261, 2011.

[3] N. X. Vinh, Coppel R. Chetty, M., and P. P. Wangikar. Globalmit: Learning globally optimal dynamic bayesian network with the mutual information test (mit) criterion doi:10.1093/bioinformatics/btr457, 2011.

[4] N. X. Vinh, Coppel R. Chetty, M., and P. P. Wangikar. Polynomial time algorithm for learning globally optimal dynamic bayesian network. *Neural Information Processing*, pages 719–729, 2011.



(a) With Stimuli



(b) Spontaneous Activity

Figure 10: Tuning Curves from the Ectorhinal Cortex

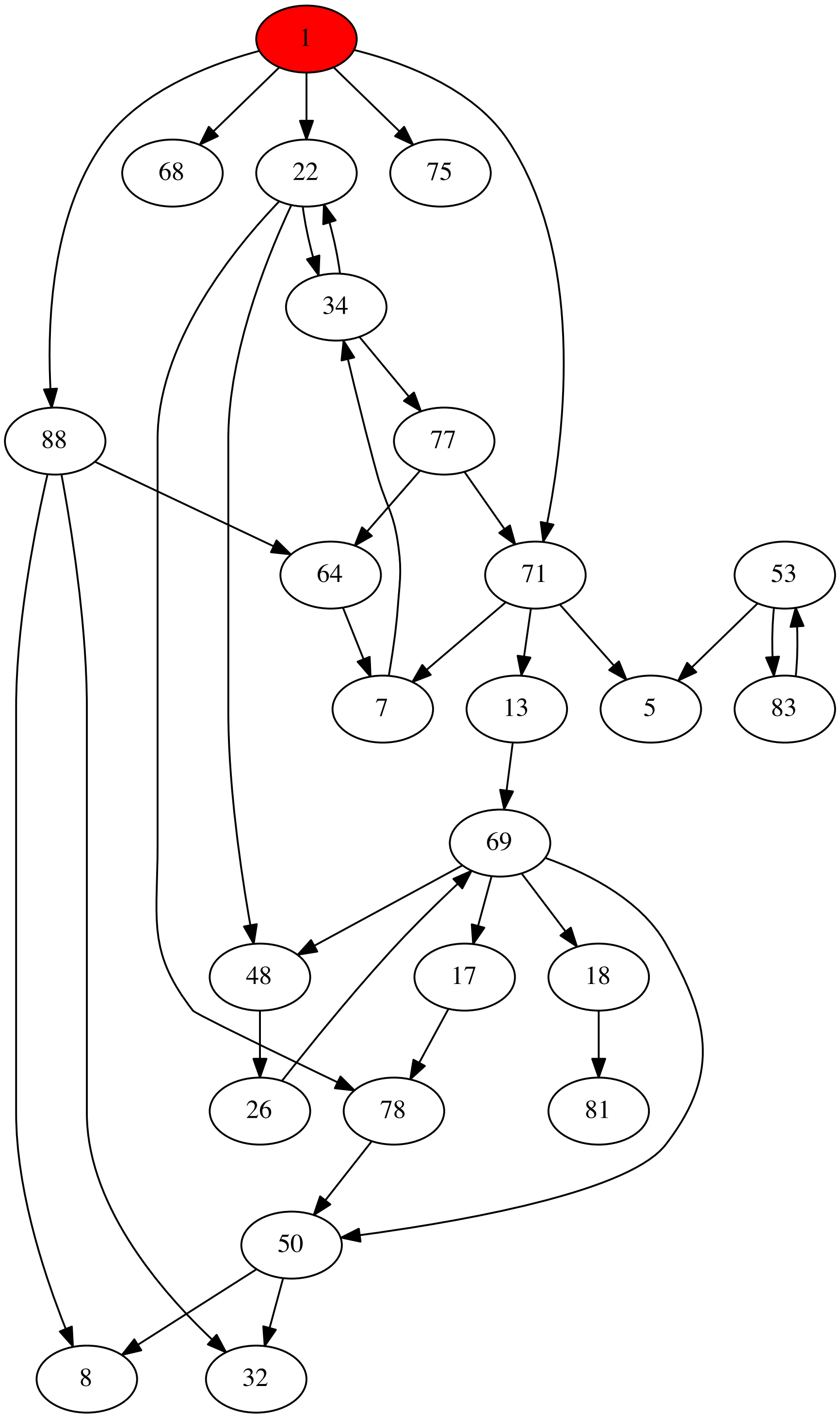
A Visualizing our Results

Here, we are including graphical representations of our results. All of these graphs have been processed after they were output by the GlobalMIT algorithm. We removed incoming edges from the source node. Unfortunately, we had no way to incorporate the a priori knowledge that no neuron can have a causal influence on the stimulus, so we had to modify our model after the fact to encode this knowledge. We also colored the source node red for the sake of clarity.

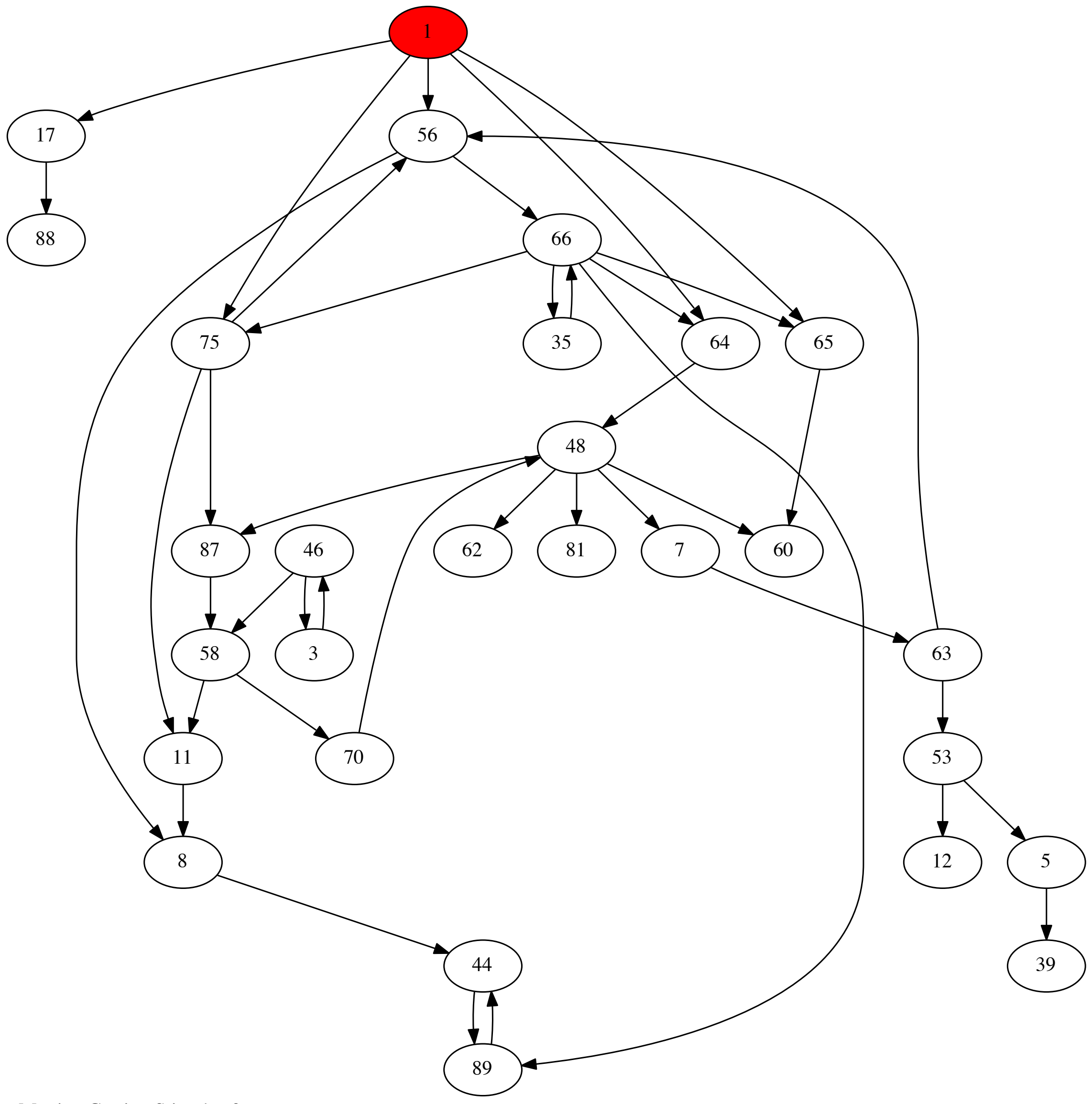
Graphs labeled “Moving Grating Stimulus” correspond to measurements in the primary visual cortex (V1) in response to gratings at specific orientations being presented to the mouse. These were our baseline — we knew what we expected to see from this data, so we used it to validate the performance of our methodology. Stimuli 1 through 16 correspond to gratings at different angles.

Graphs labeled “BL Looming Object” correspond to measurements in the ectorhinal cortex without the prescence of a stimulus, random activity. These allowed for us to measure the effect of a looming object stimulus on the neural activity in the region compared to random activity.

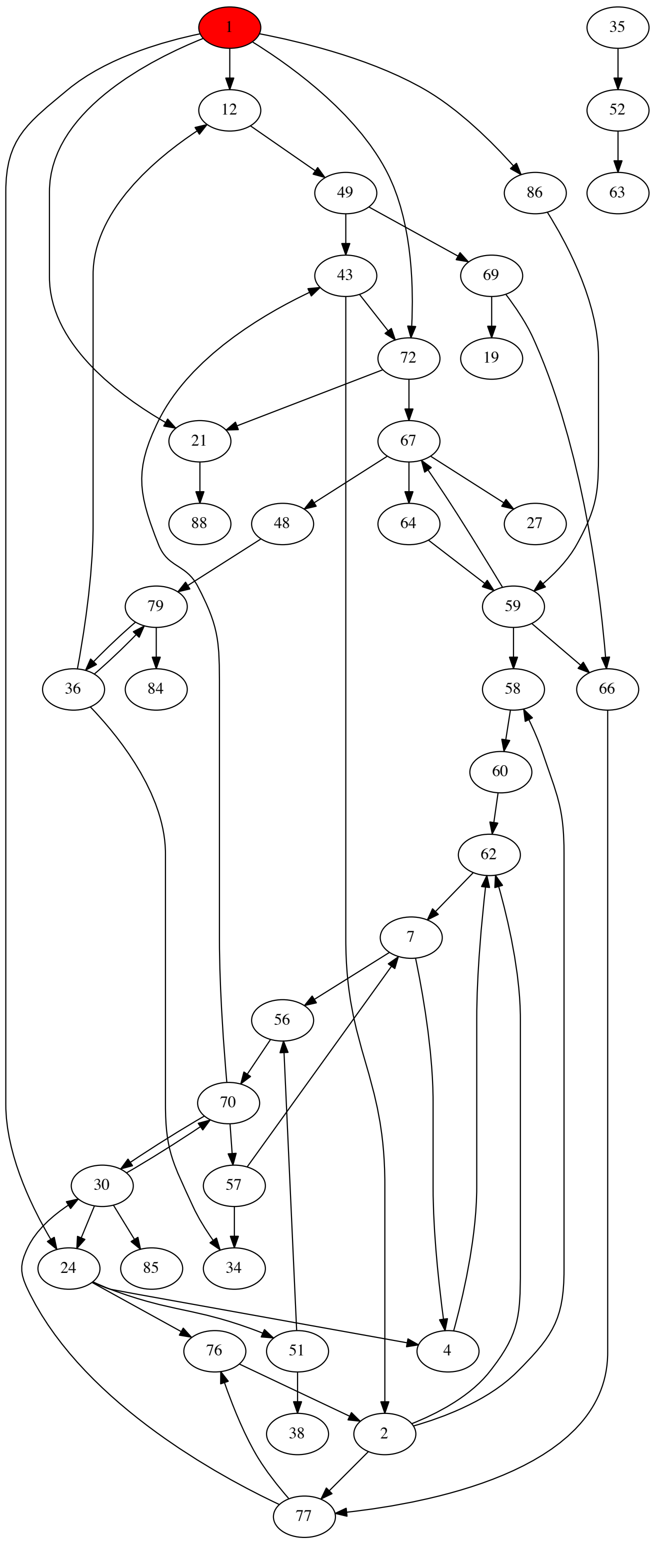
Graphs labeled “Looming Object” correspond to measurements in the ectorhinal cortex during the presentation of novel stimuli. This was the dataset that we were most interested in exploring. Stimuli 1 through 16 correspond to stimuli of different shapes.



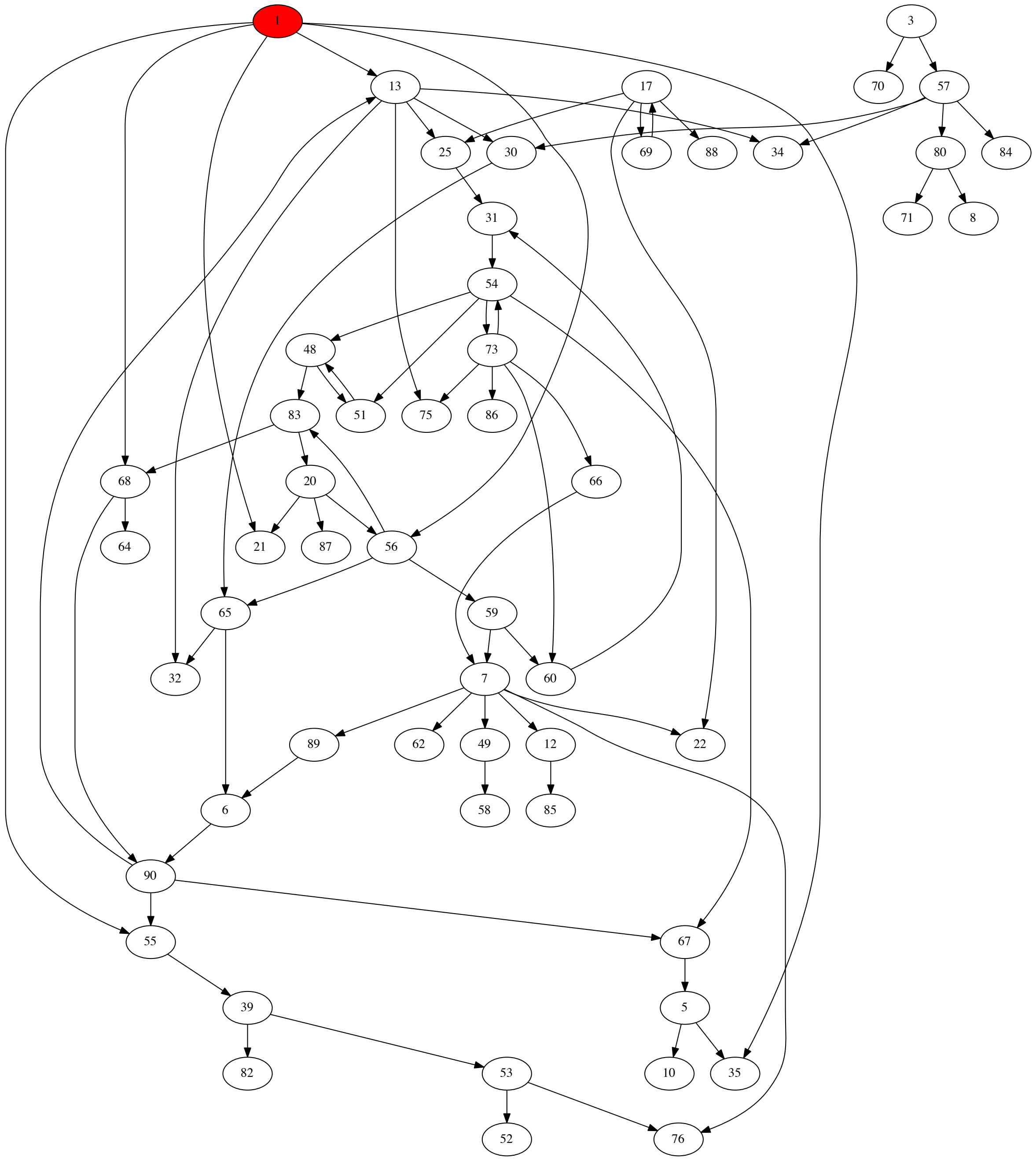
Moving Grating Stimulus 1



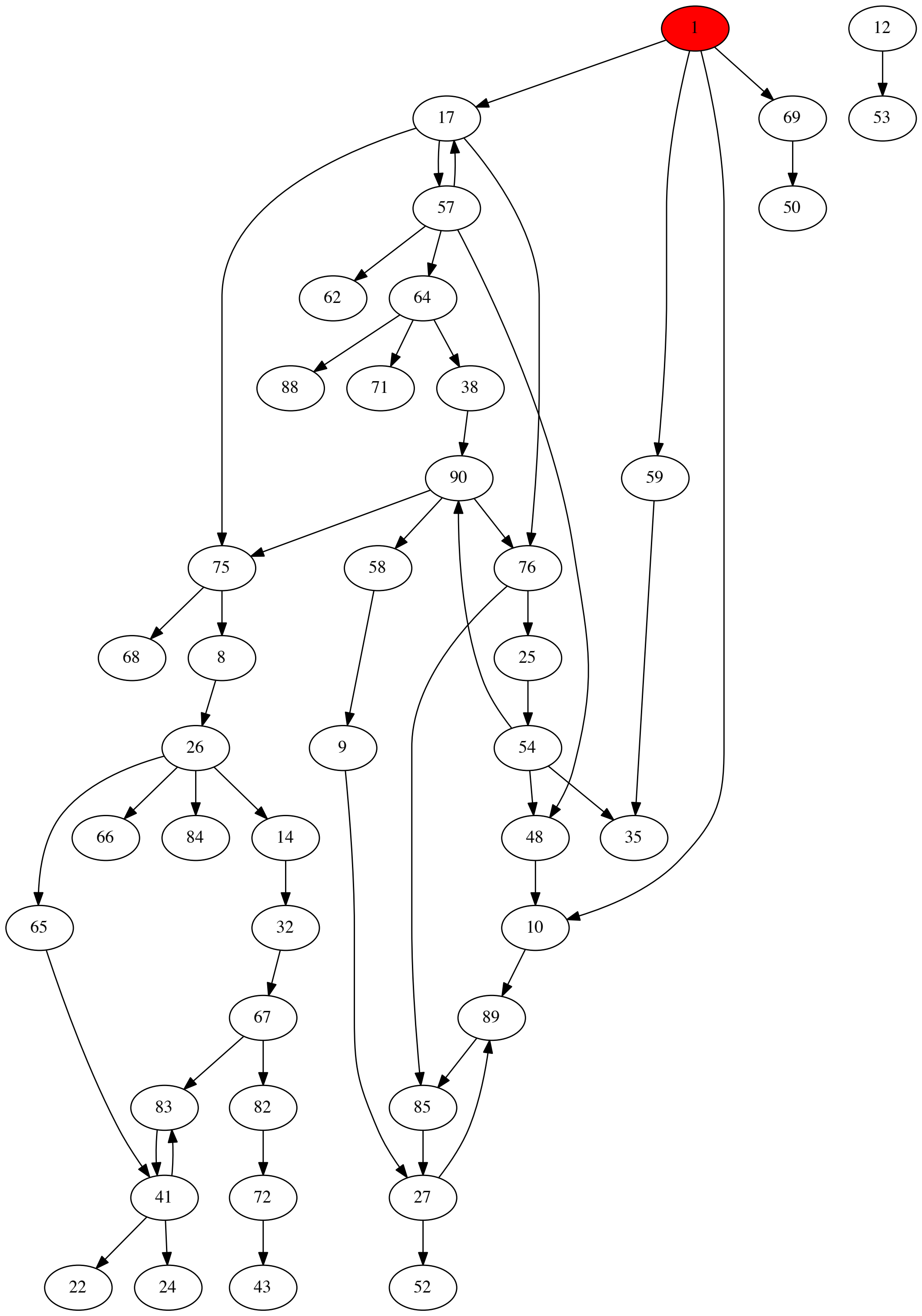
Moving Grating Stimulus 2



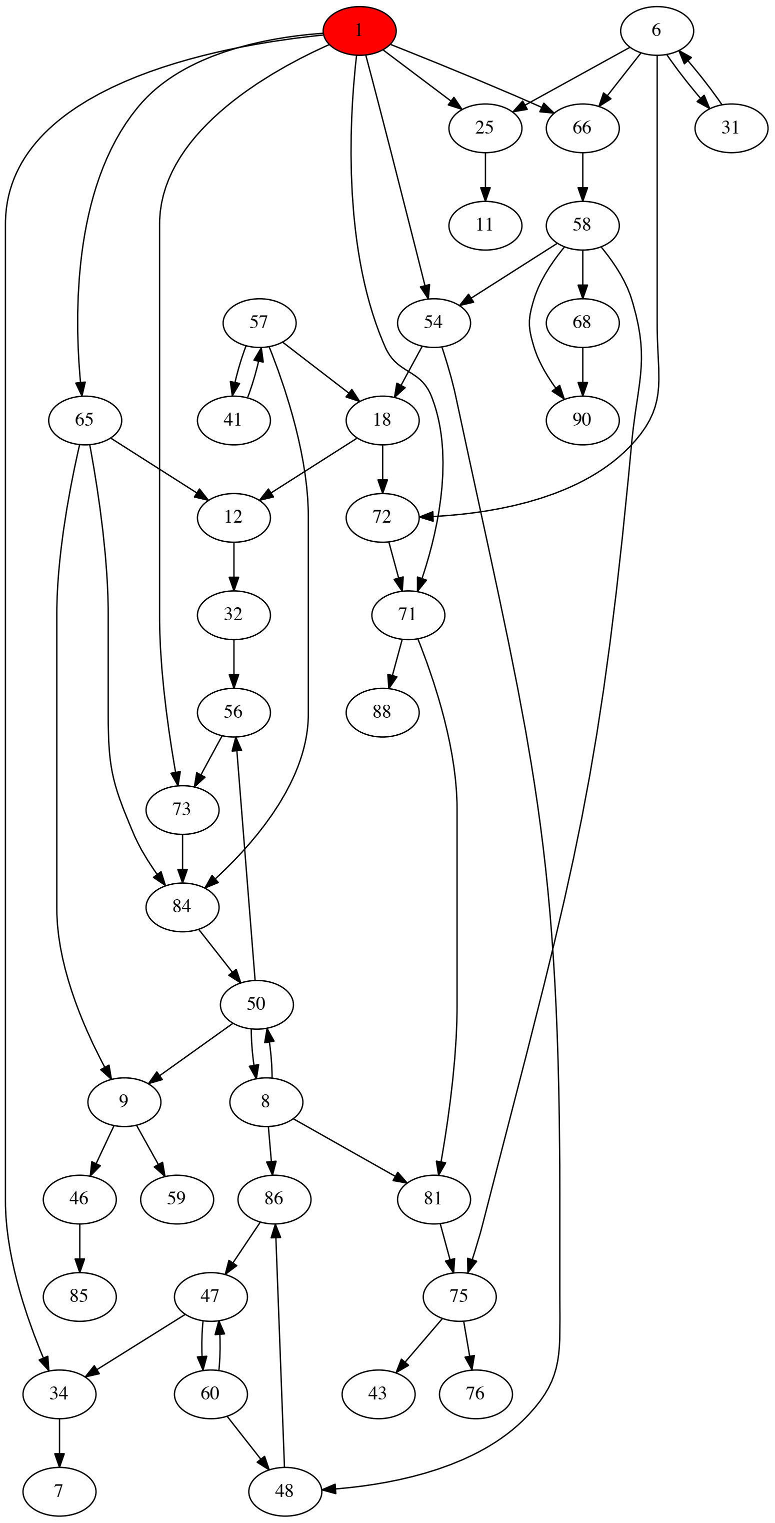
Moving Grating Stimulus 3



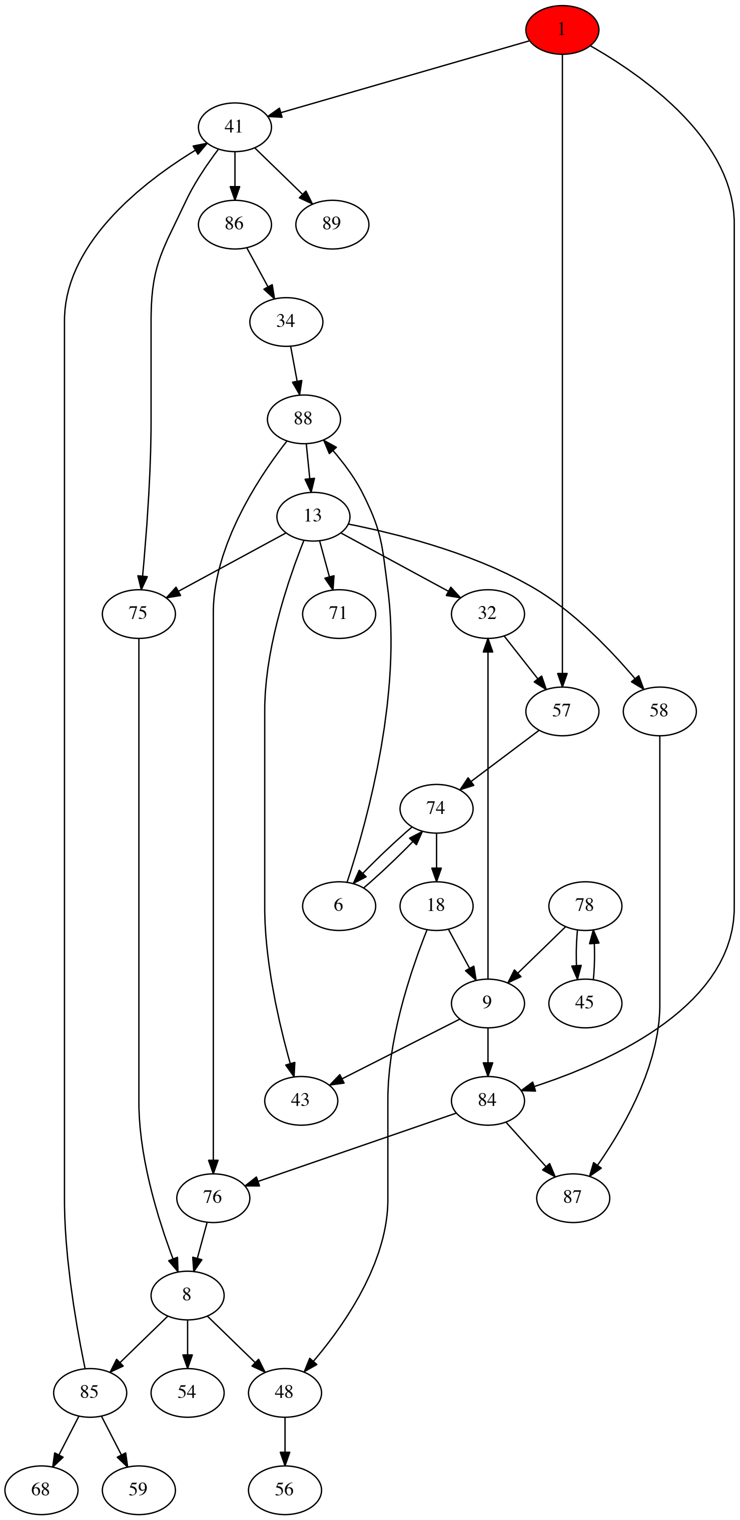
Moving Grating Stimulus 4



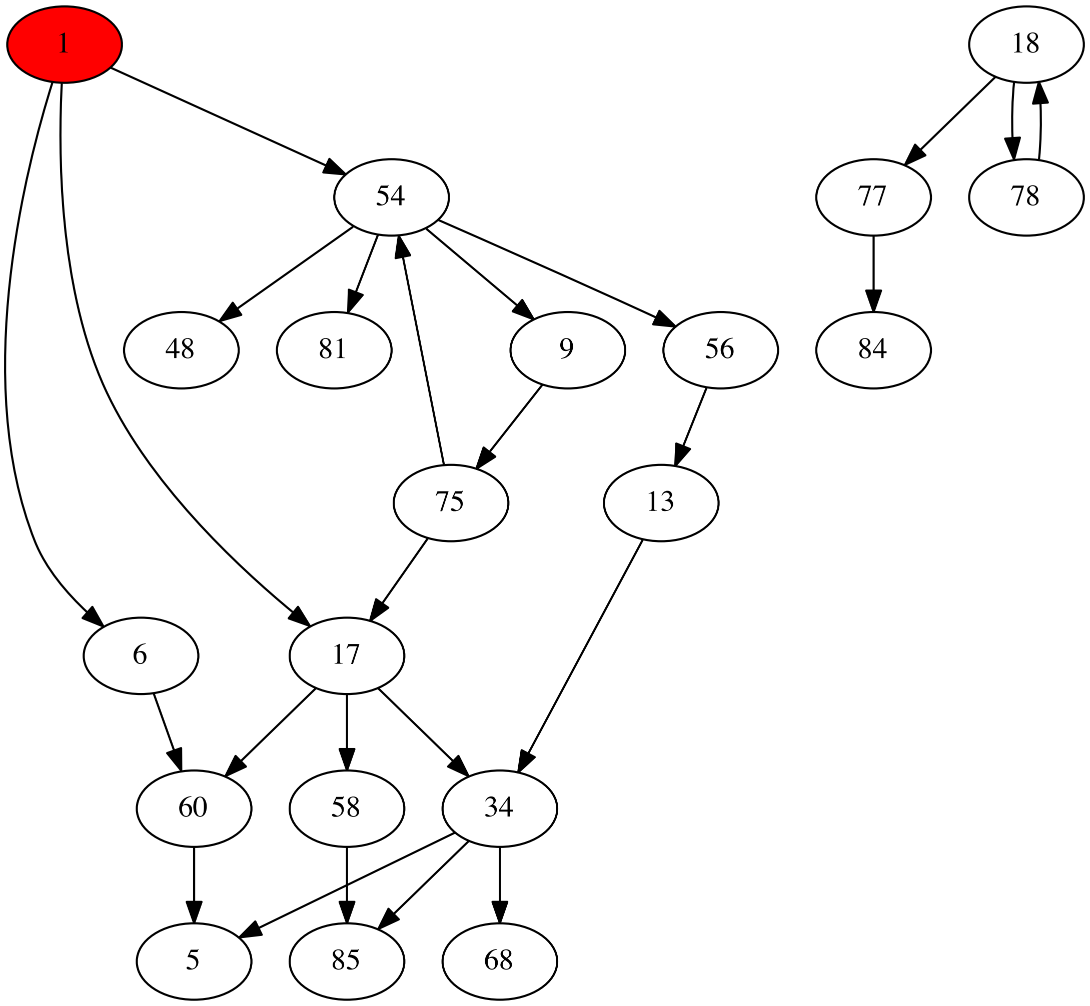
Moving Grating Stimulus 5



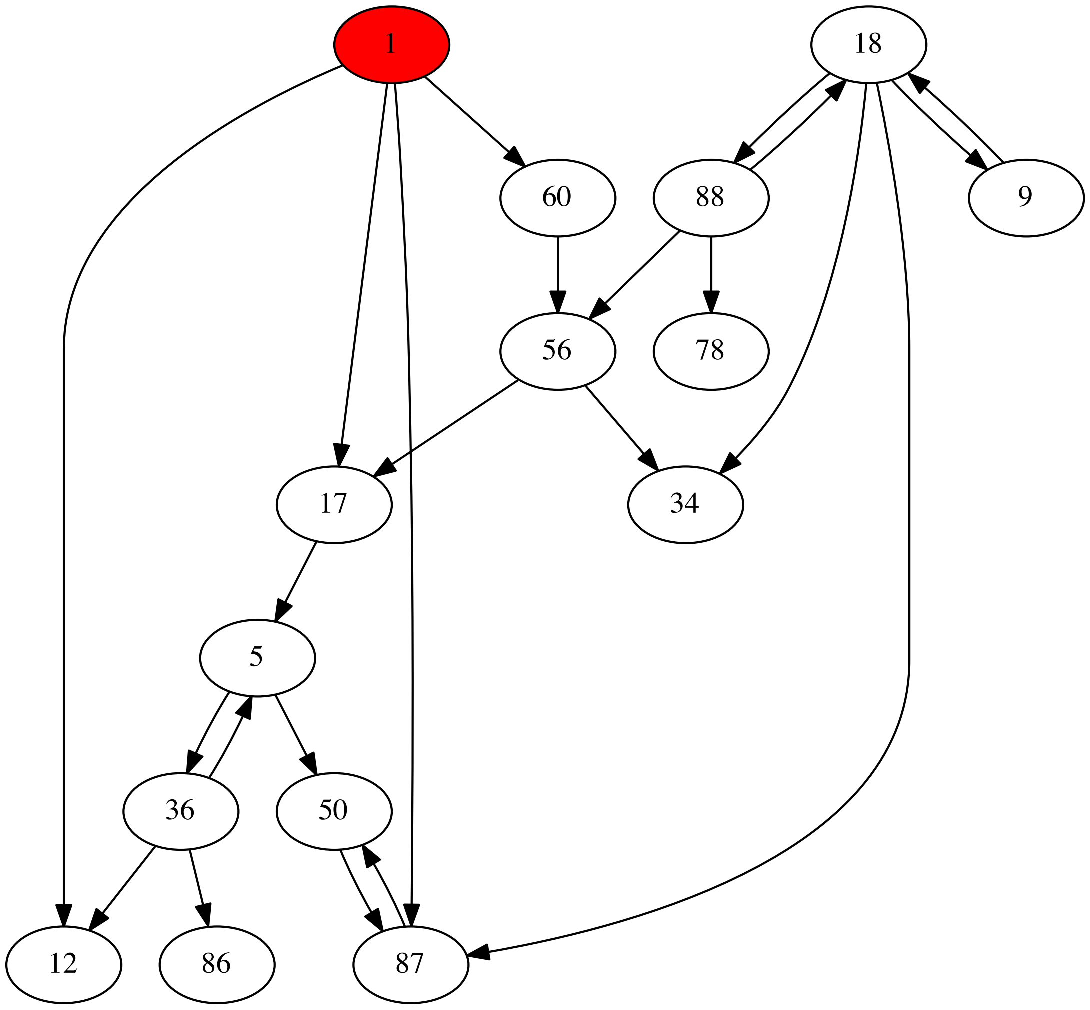
Moving Grating Stimulus 6



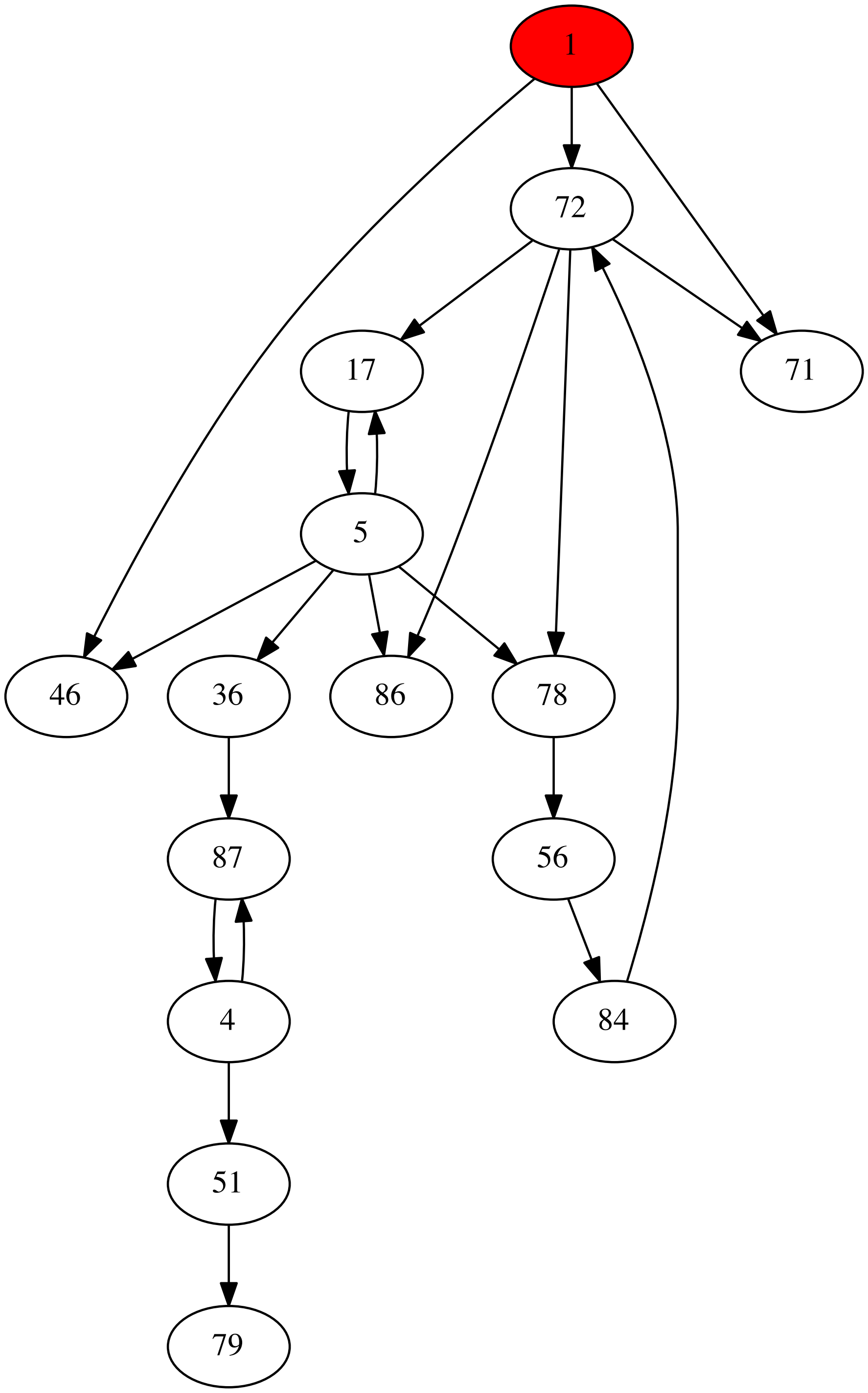
Moving Grating Stimulus 7



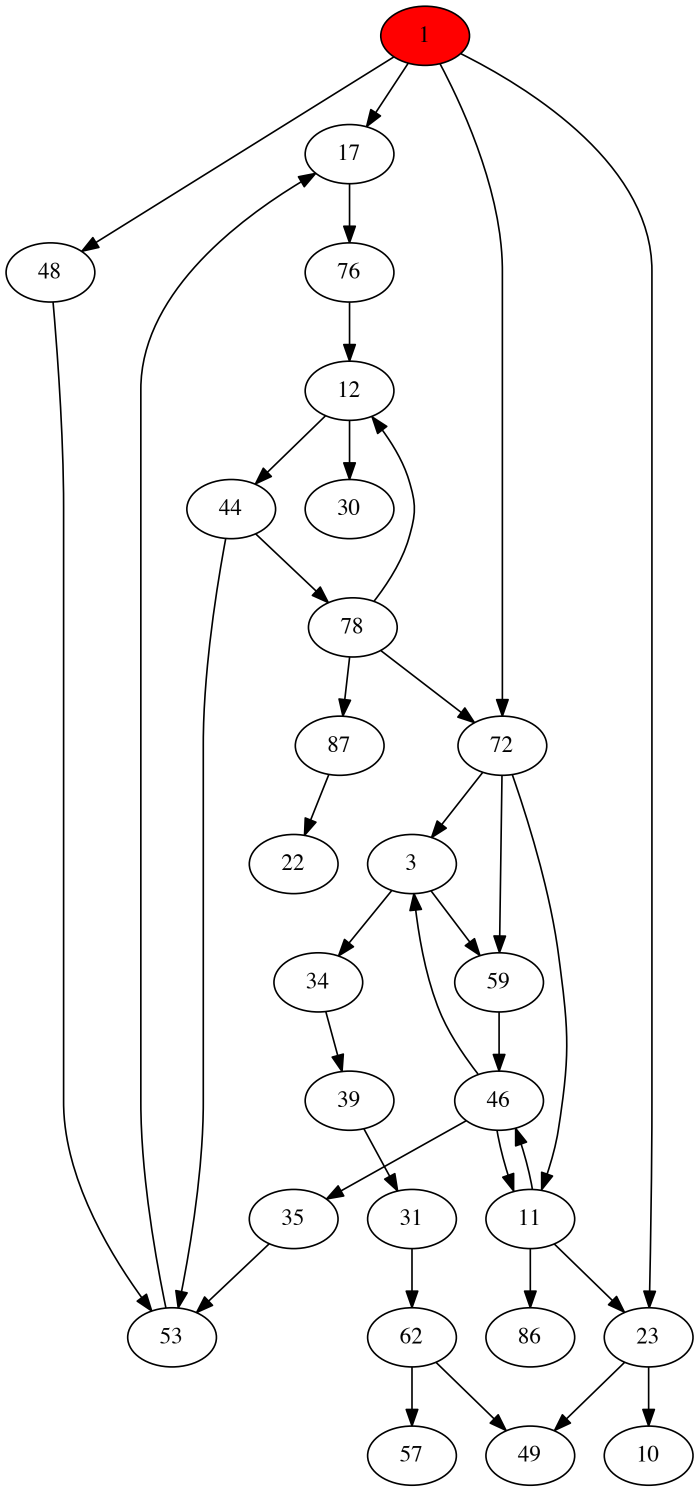
Moving Grating Stimulus 8



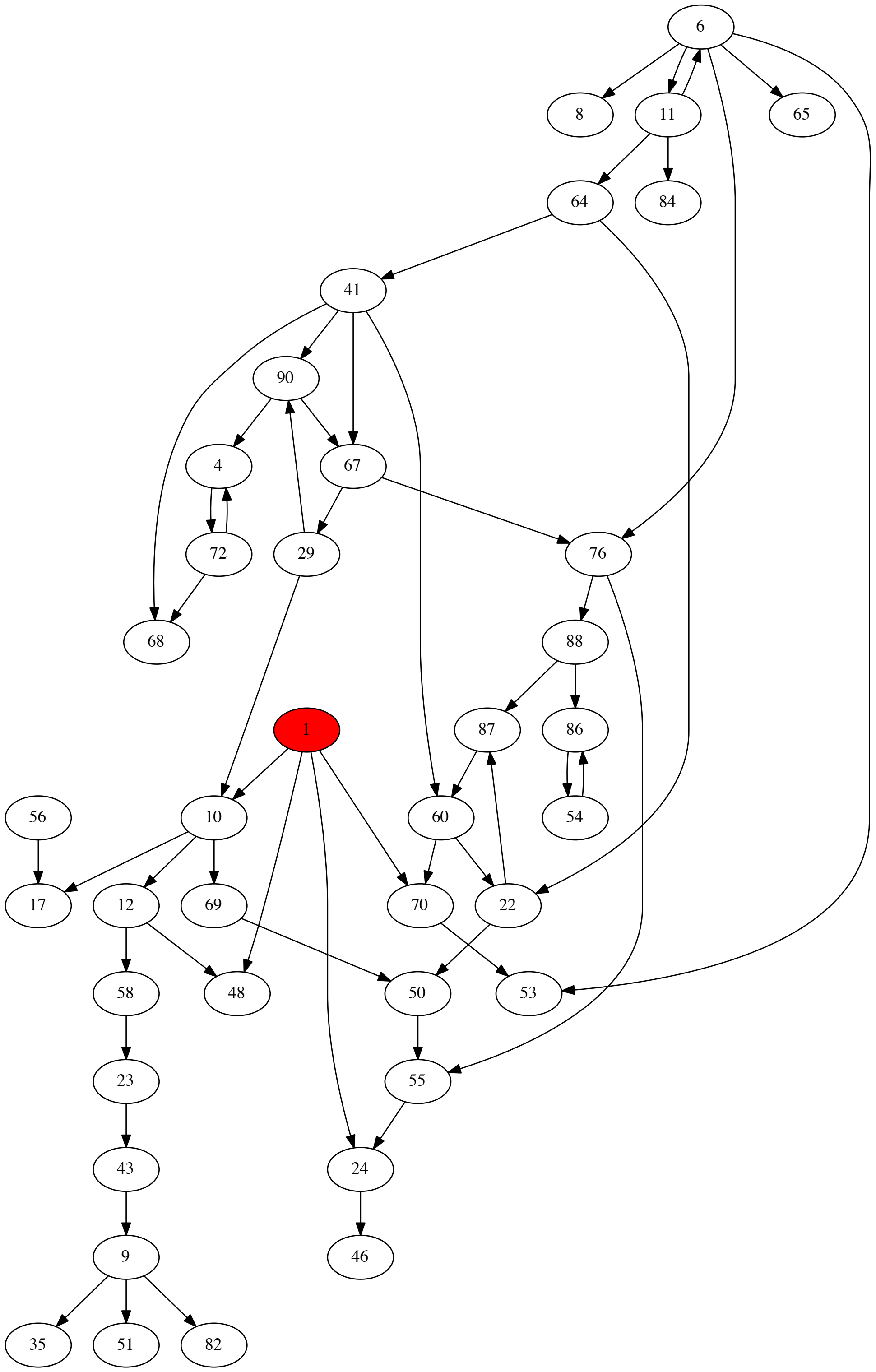
Moving Grating Stimulus 9



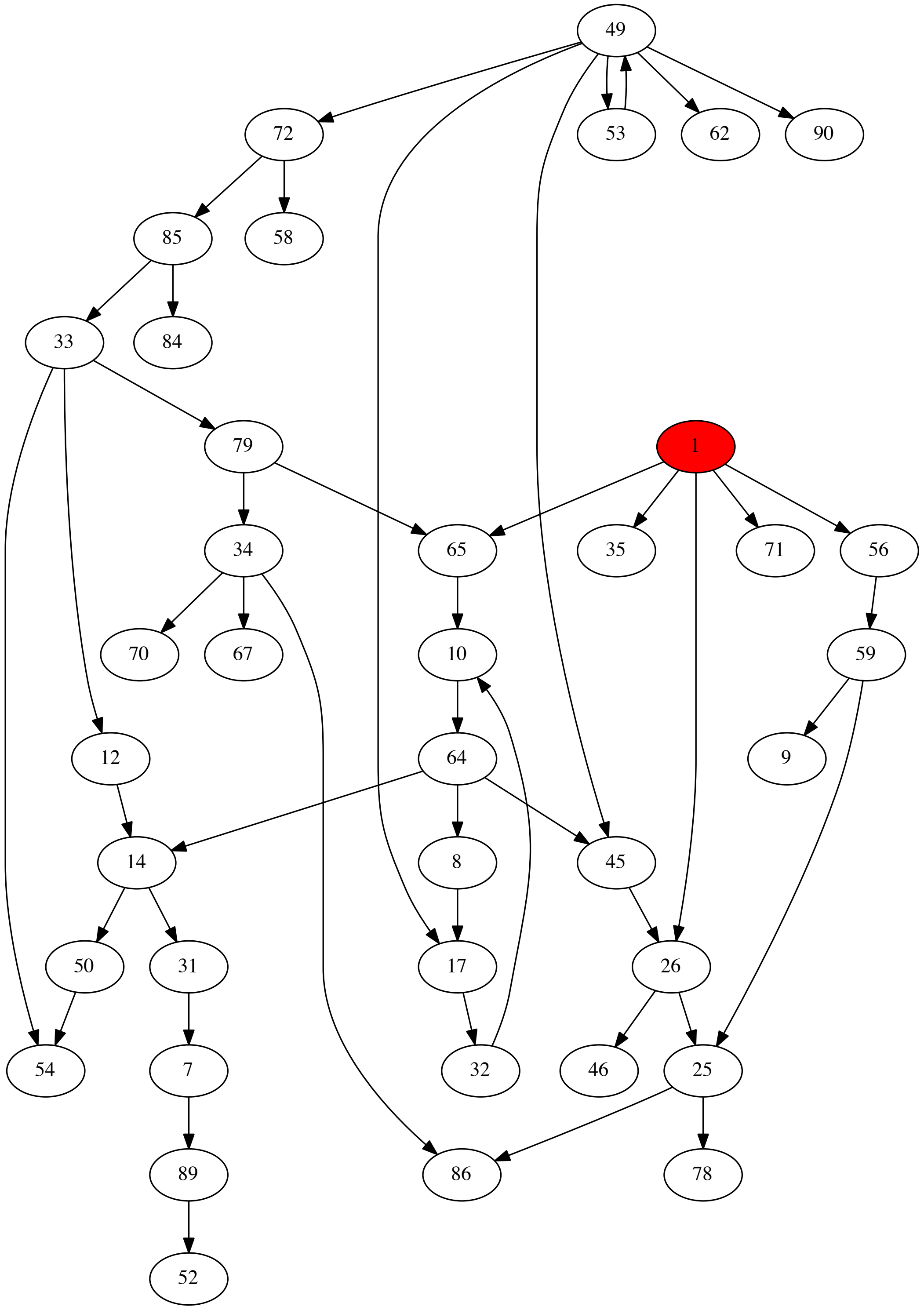
Moving Grating Stimulus 10



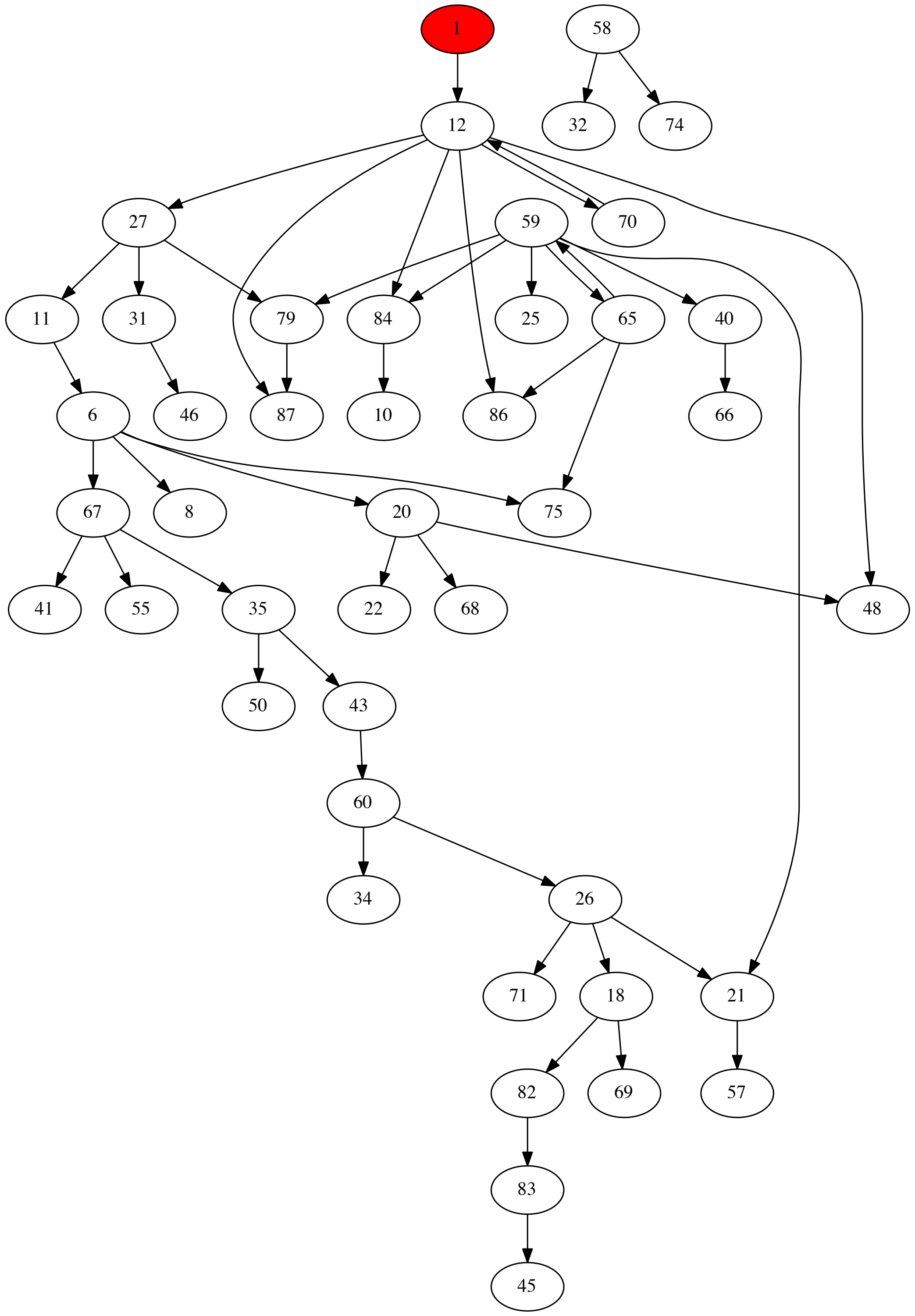
Moving Grating Stimulus 11



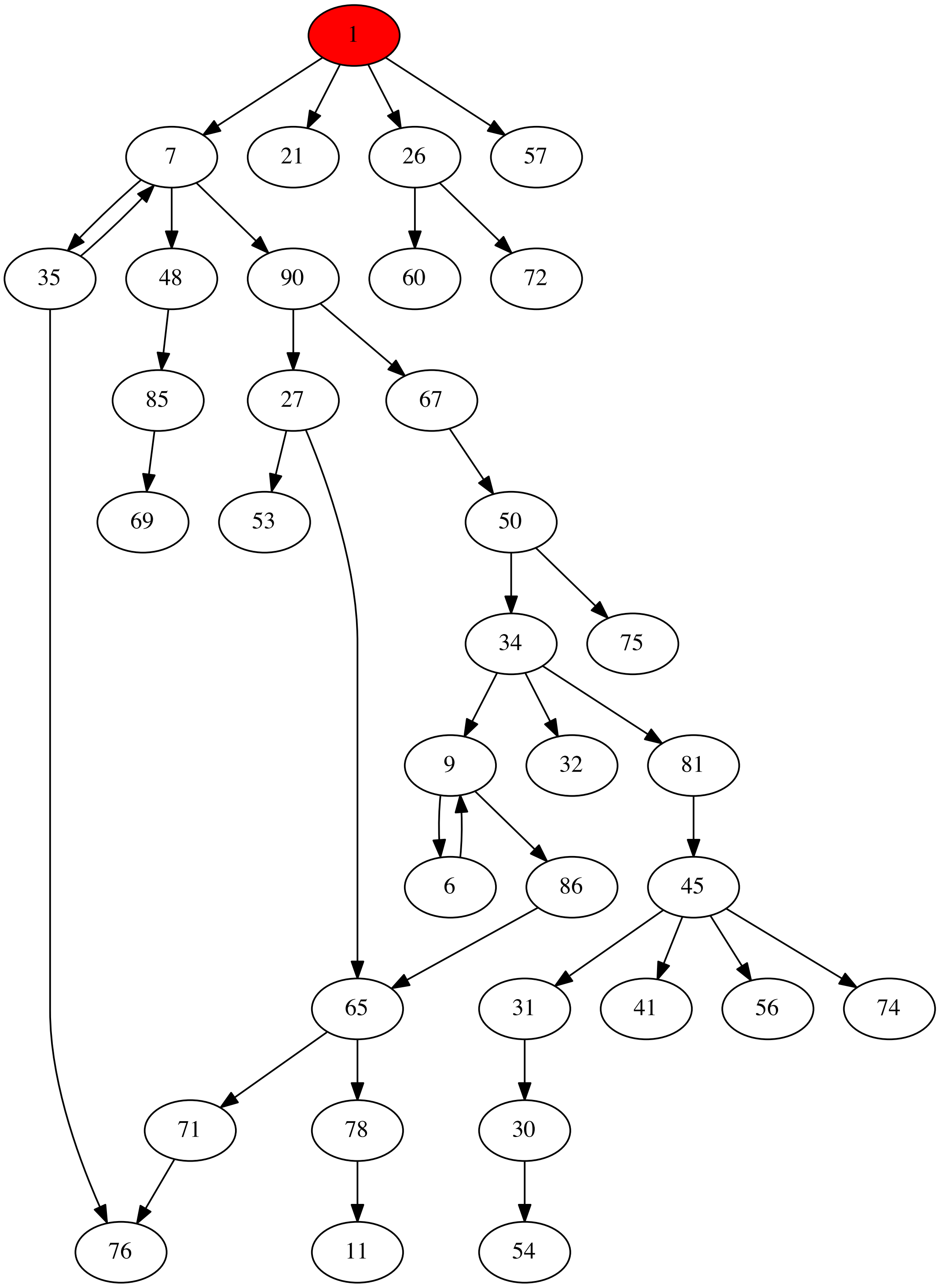
Moving Grating Stimulus 12



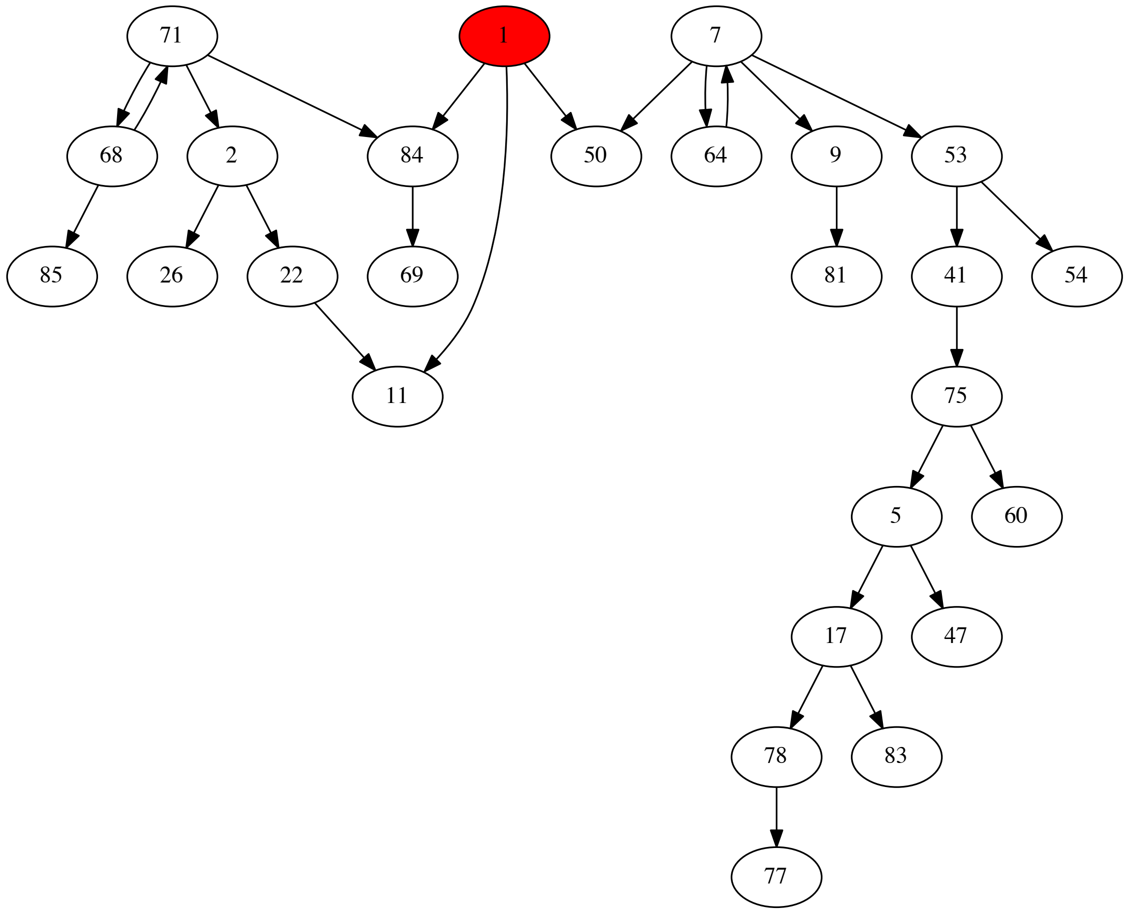
Moving Grating Stimulus 13



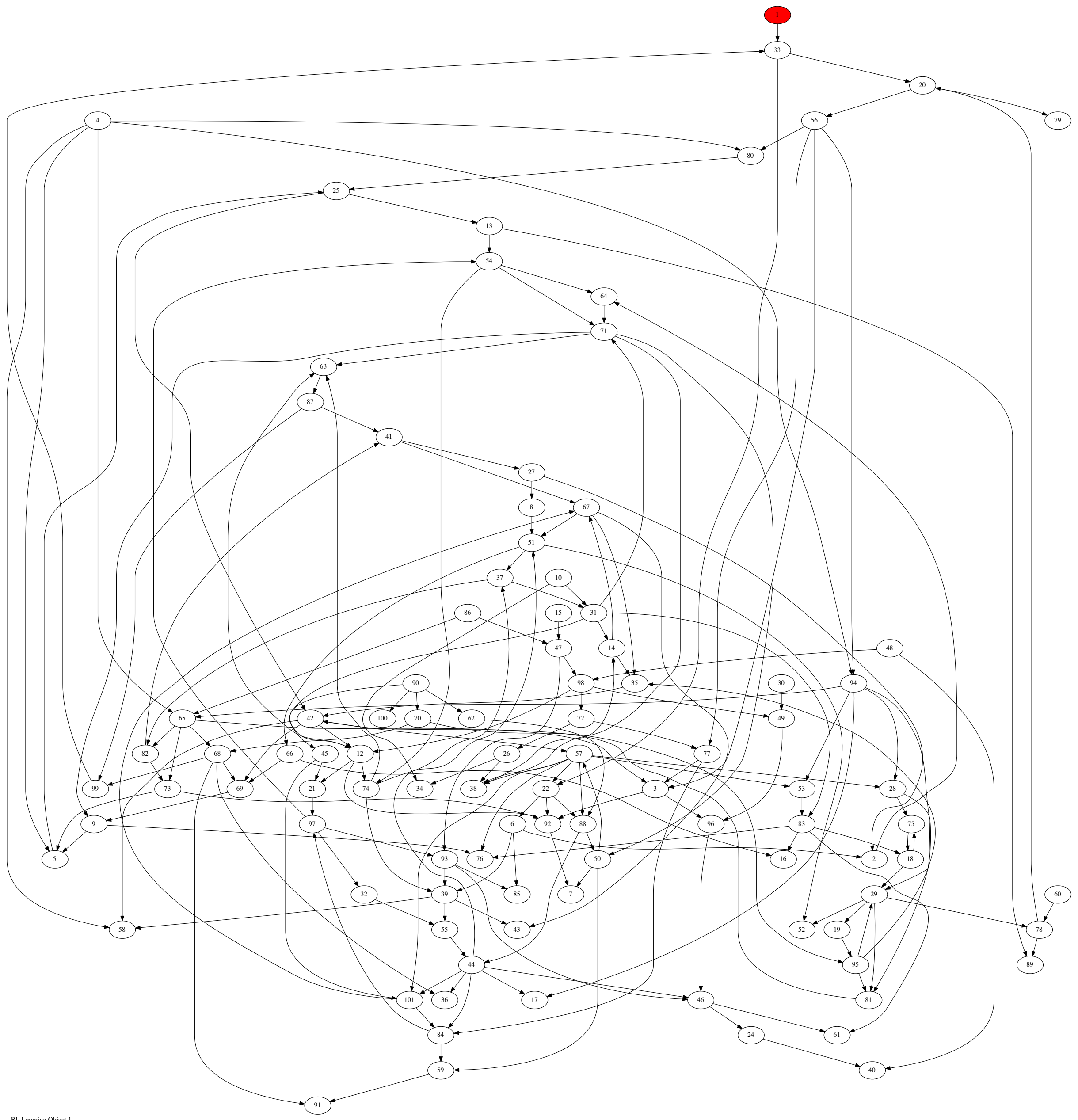
Moving Grating Stimulus 14



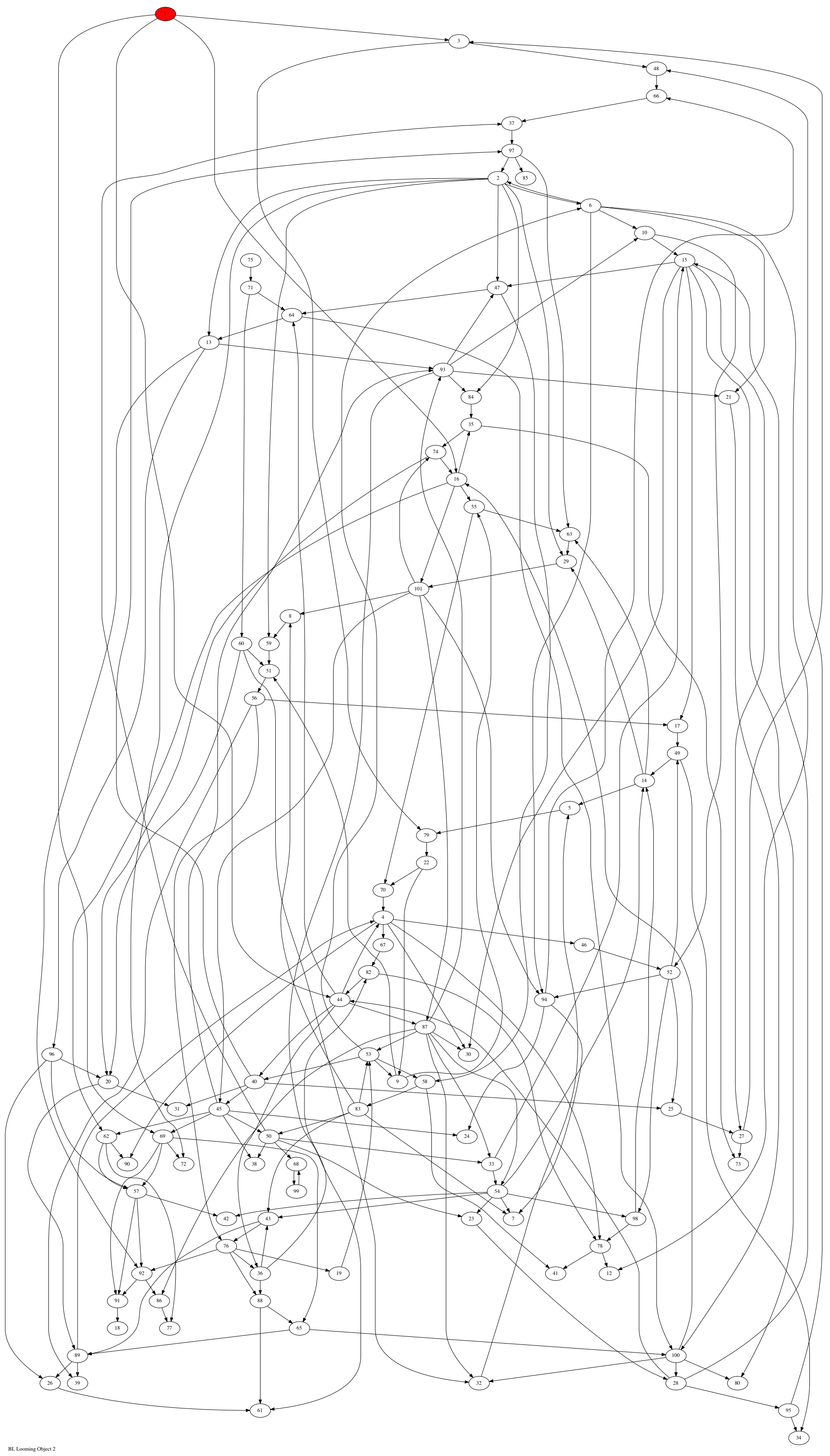
Moving Grating Stimulus 15



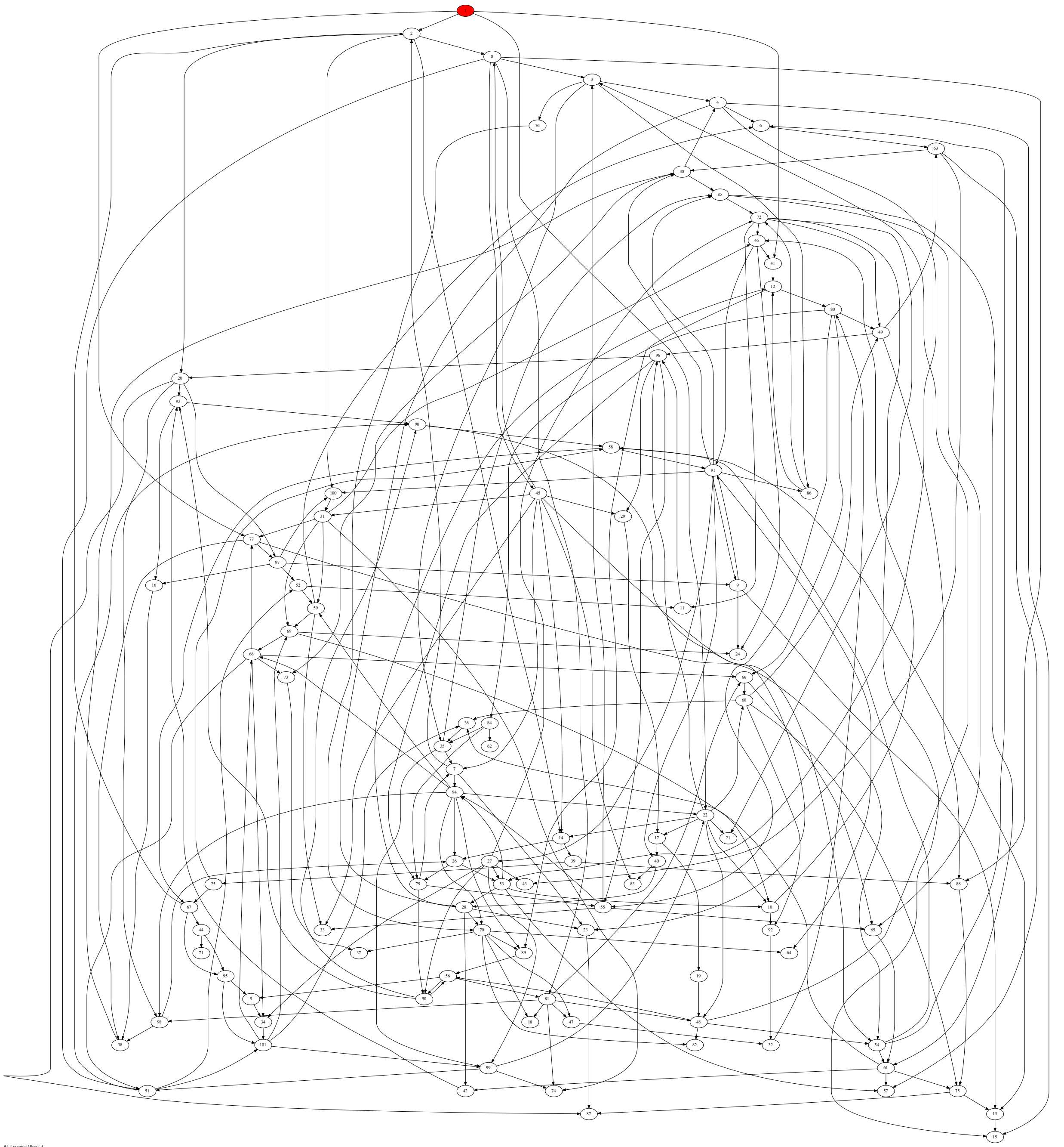
Moving Grating Stimulus 16



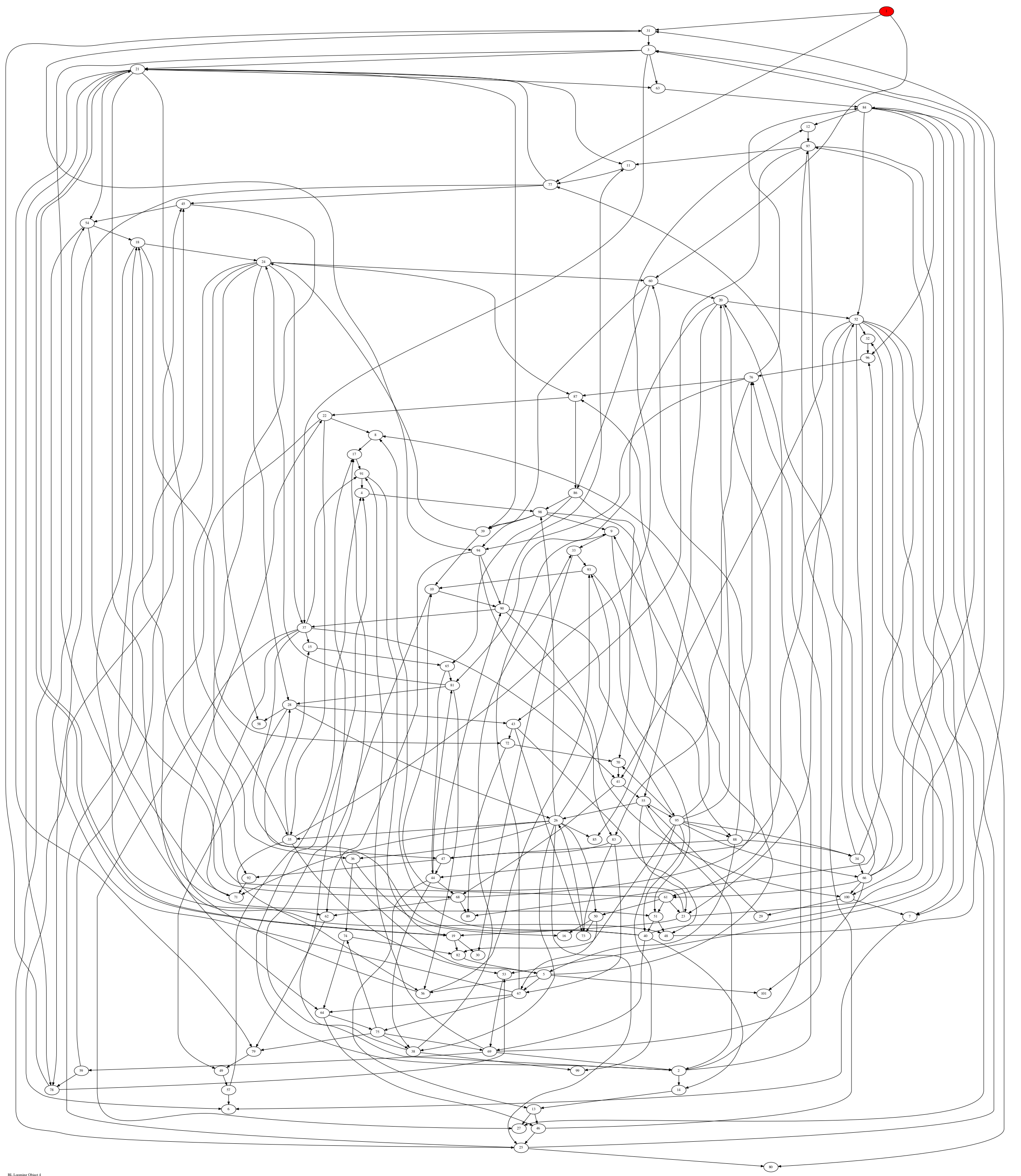
BL Looming Object 1



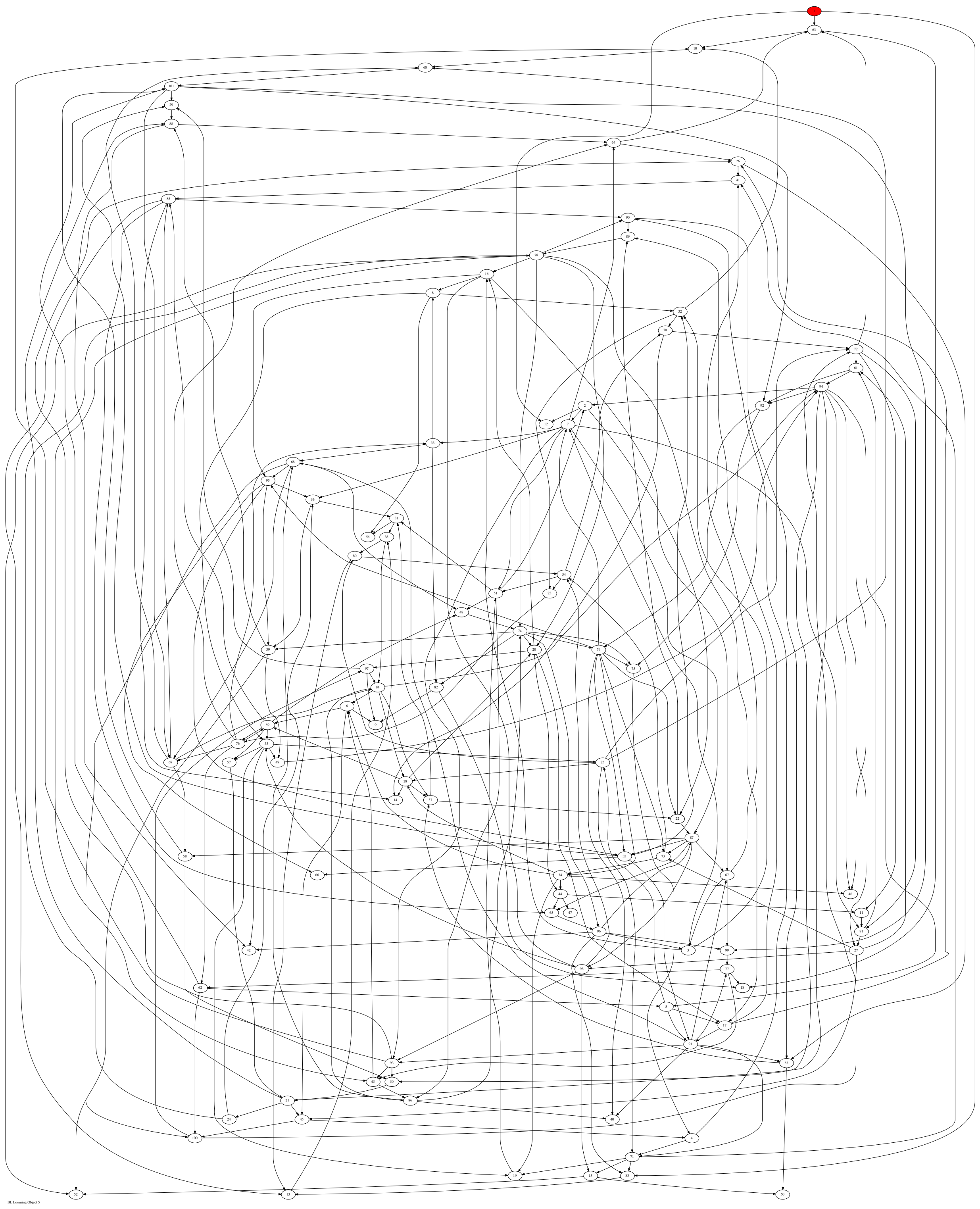
BL Looming Object 2



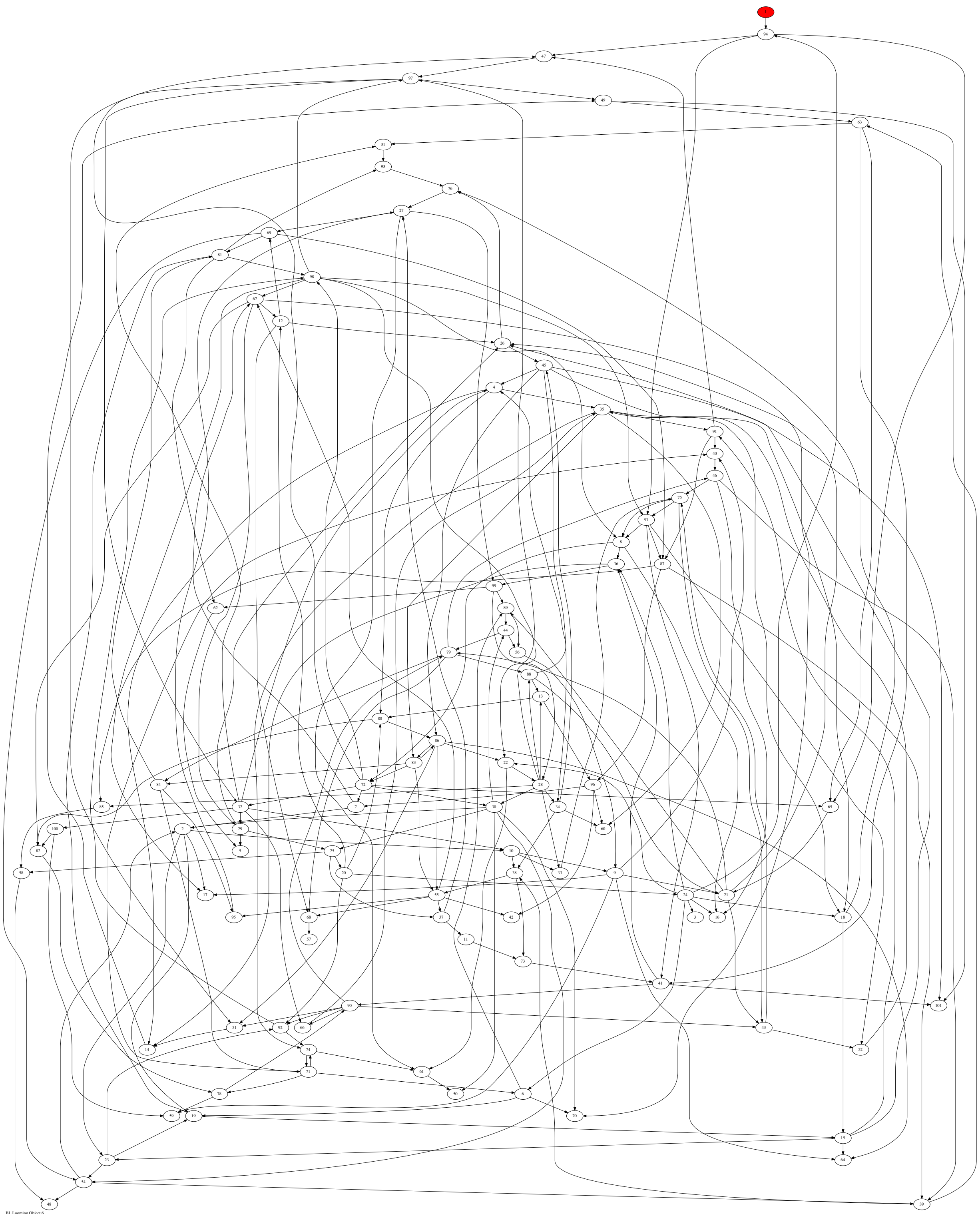
BL Looming Object 3

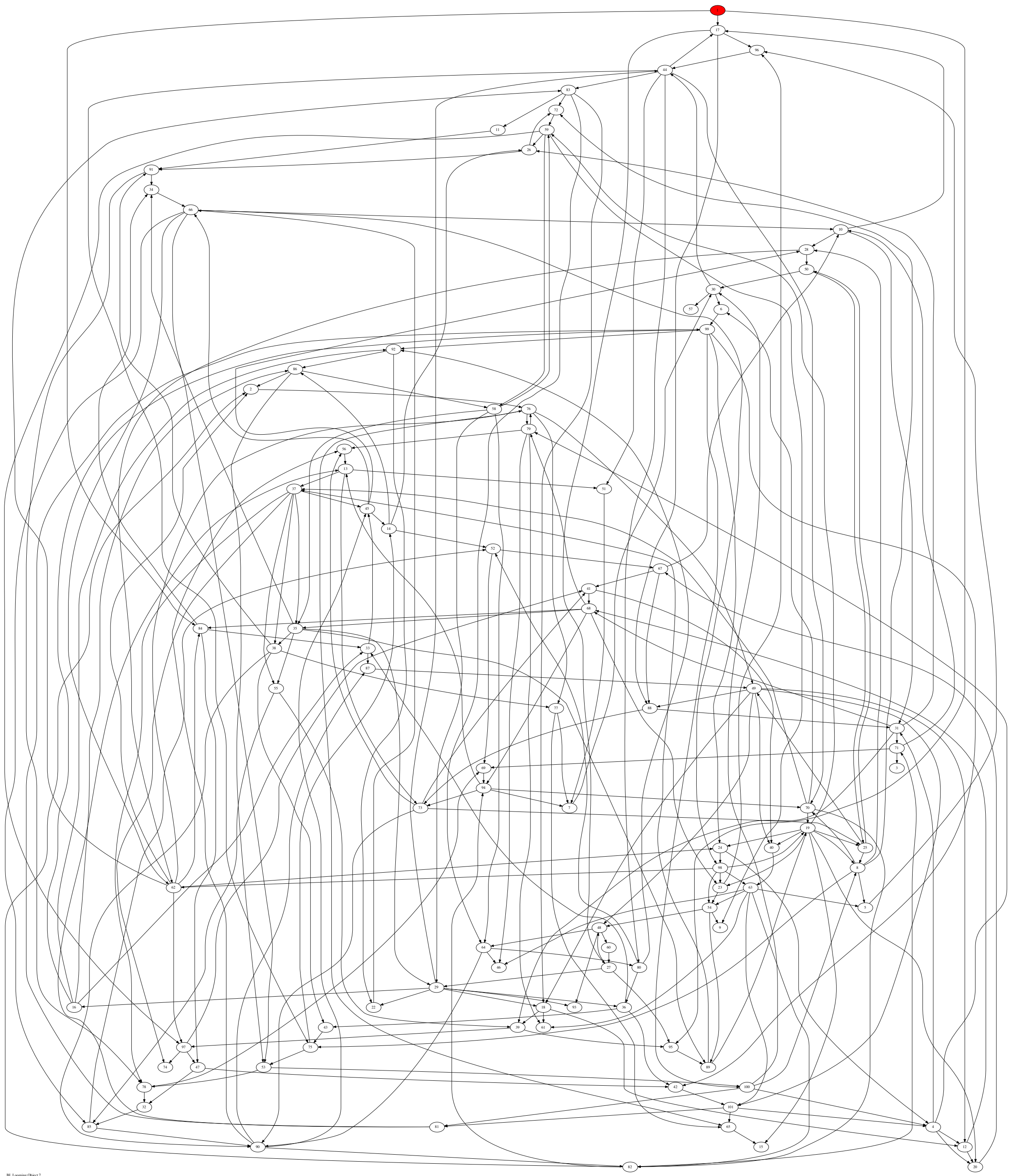


BL Looming Object 4

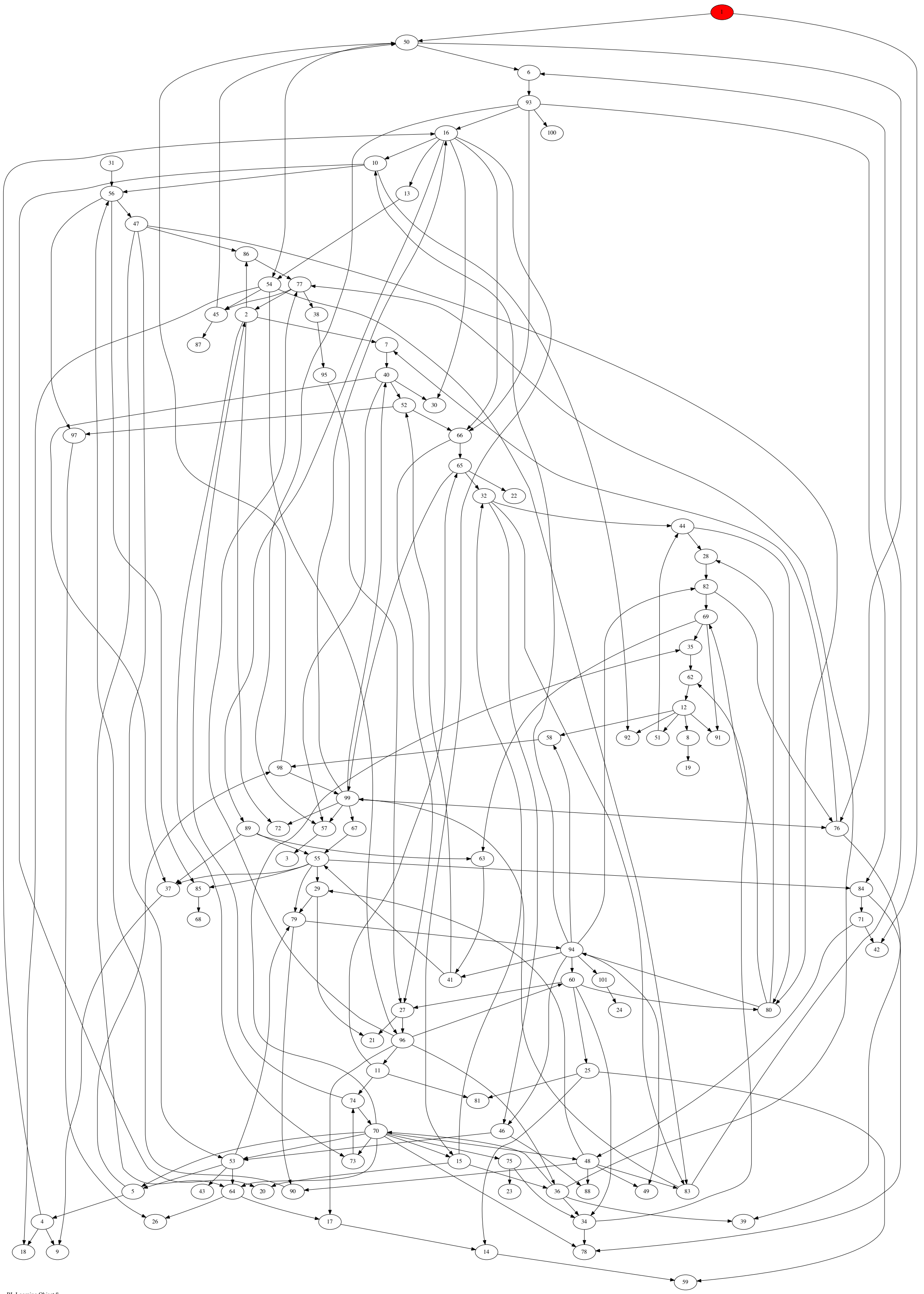


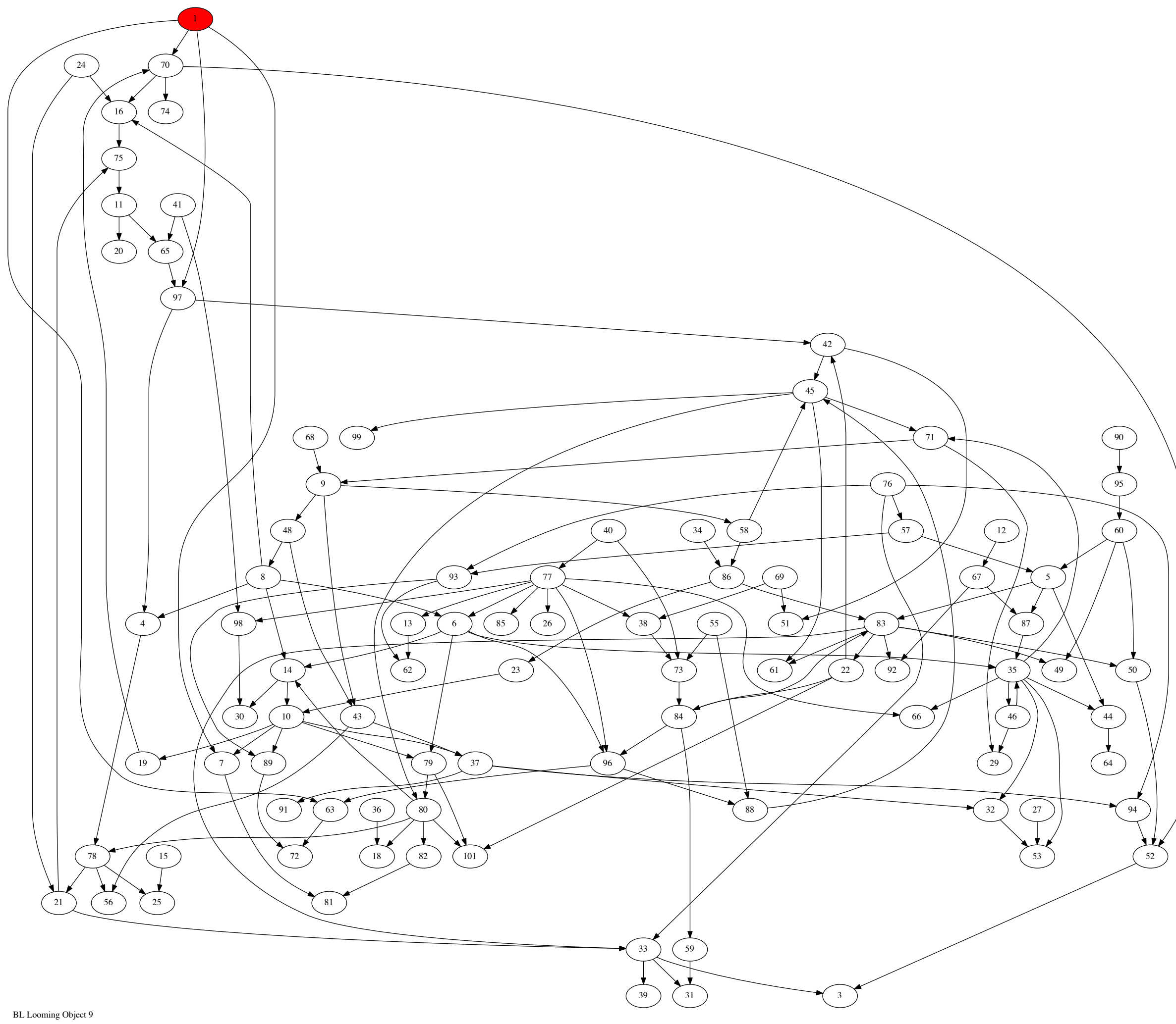
BL Looming Object 5


BL Looming Object 6

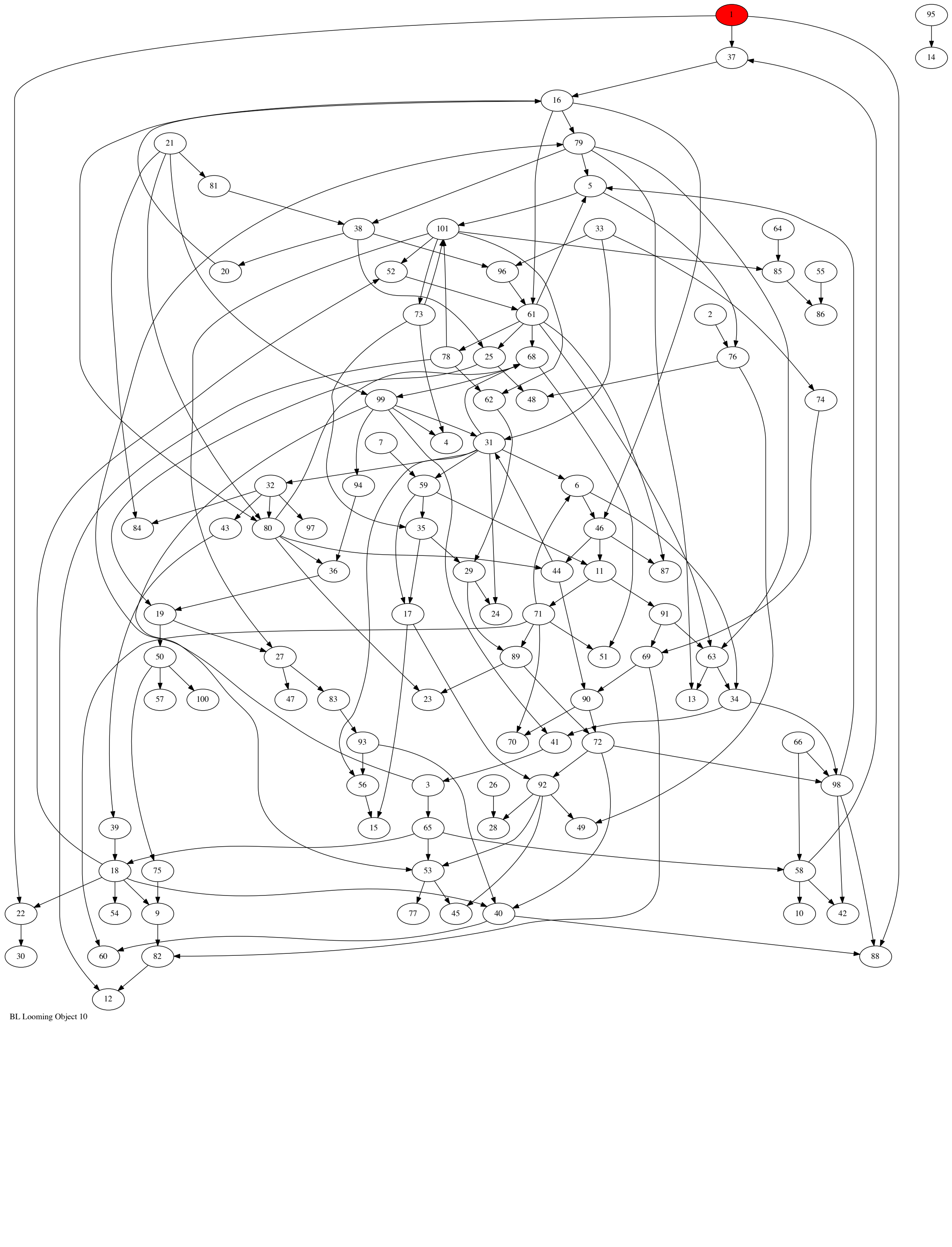


BL Looming Object 7

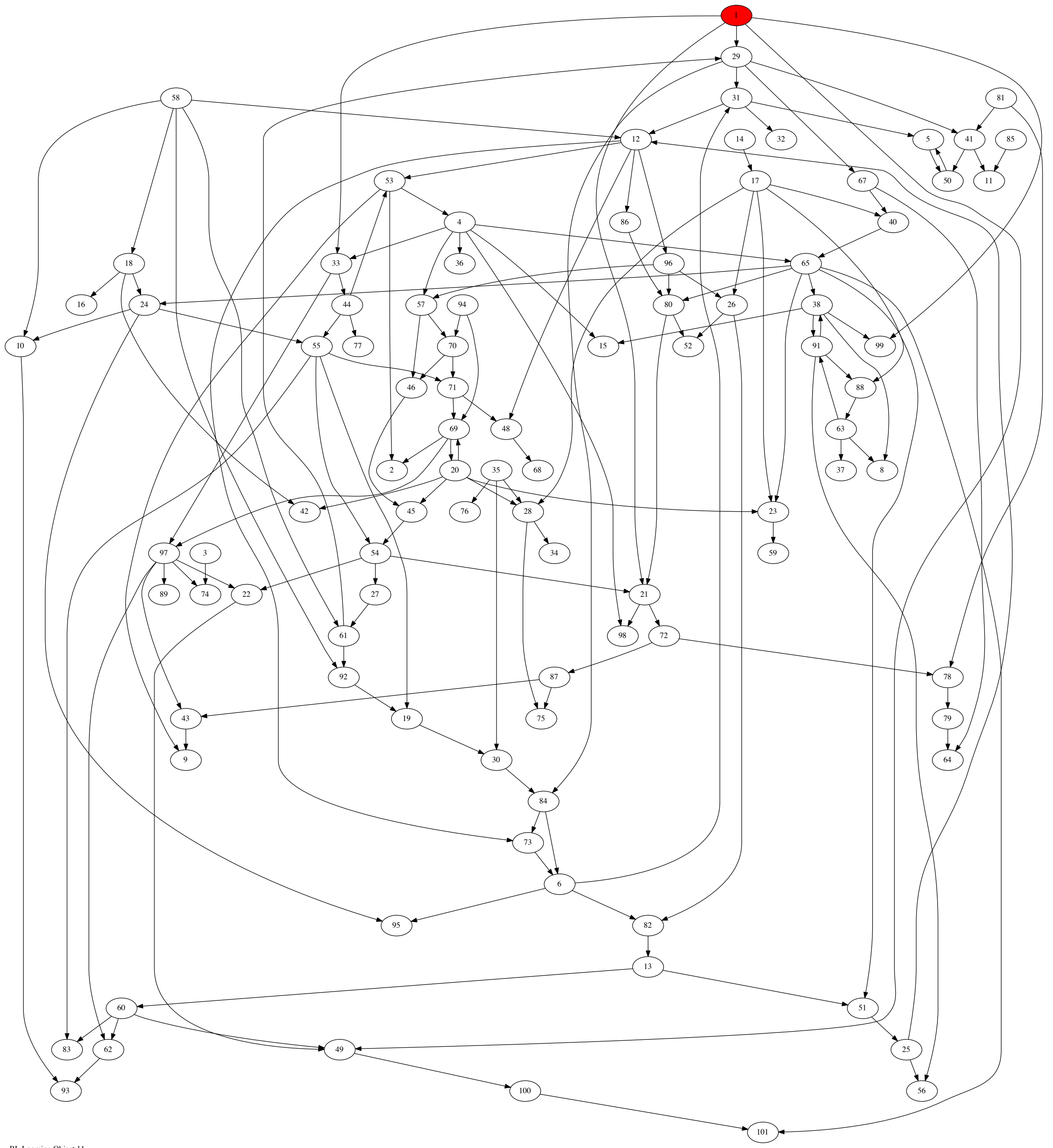

BL Looming Object 8



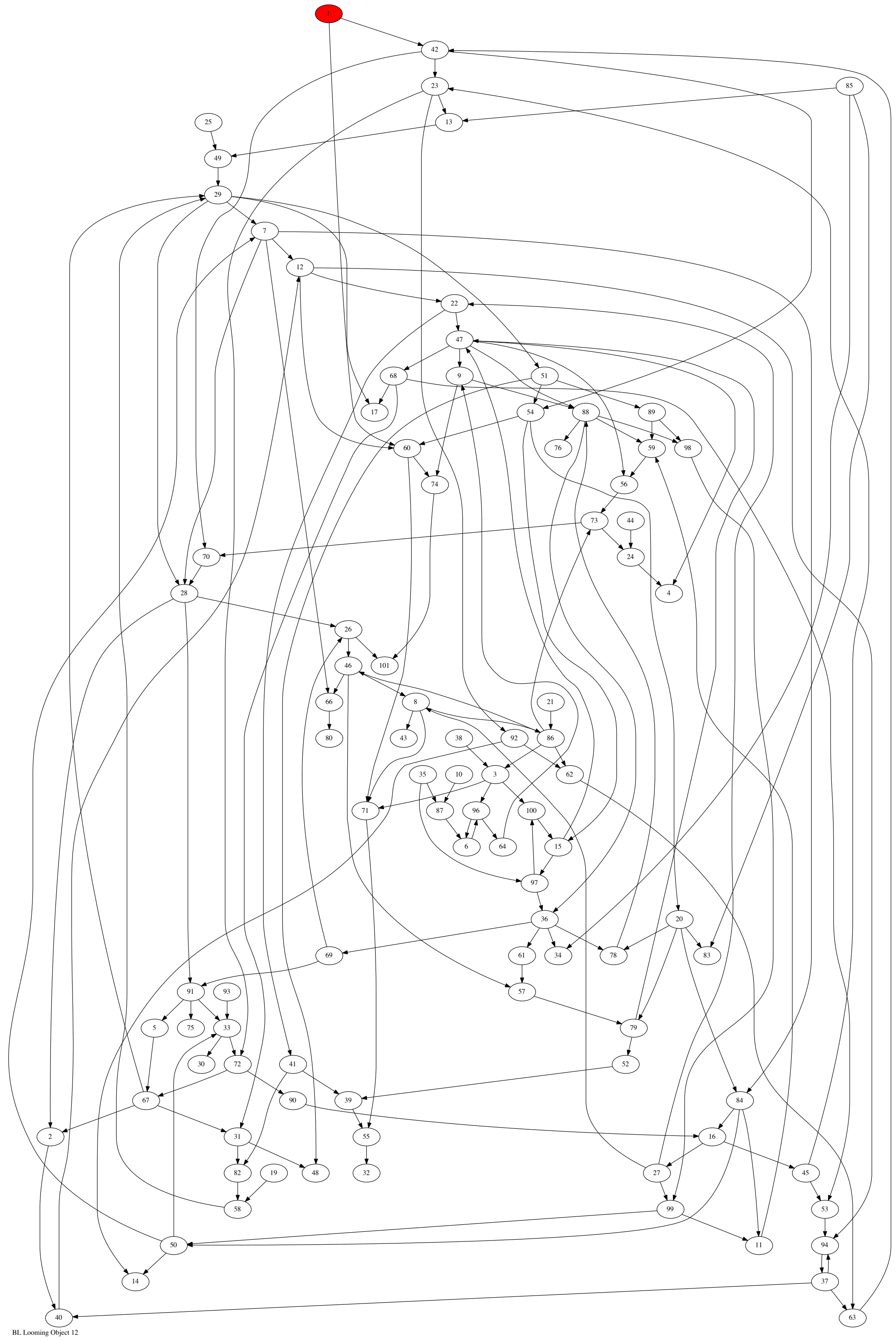
BL Looming Object 9



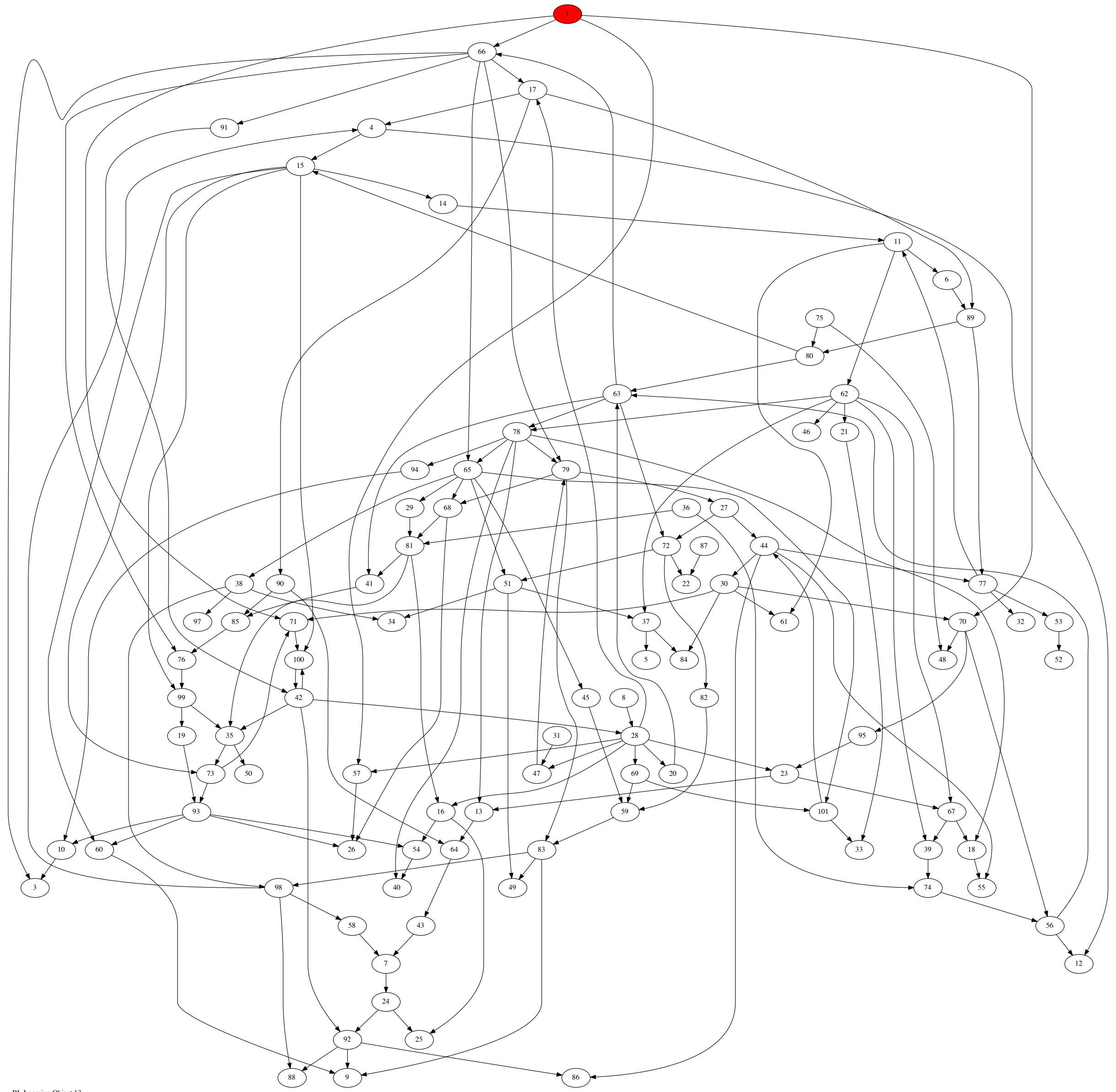
BL Looming Object 10

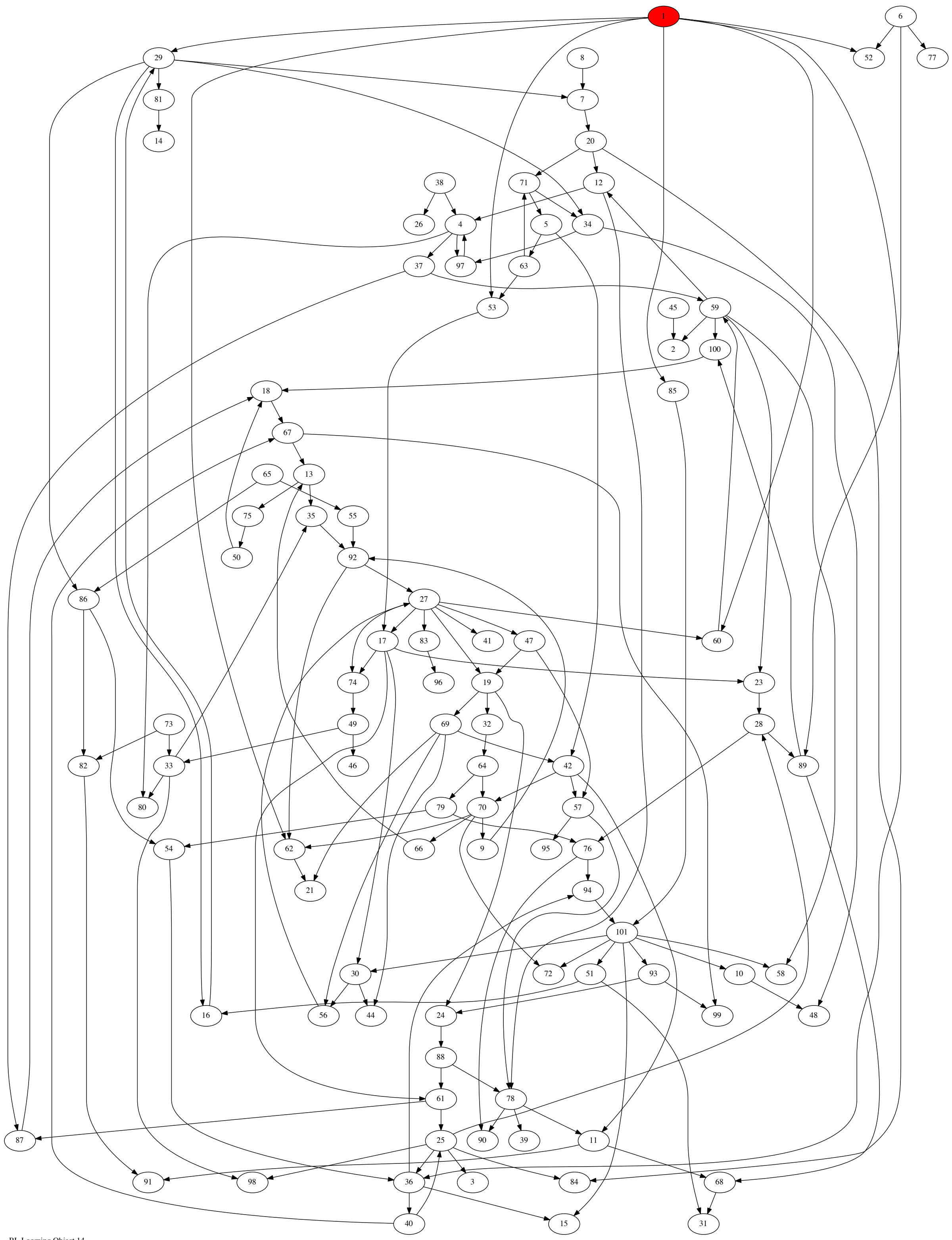


BL Looming Object 11

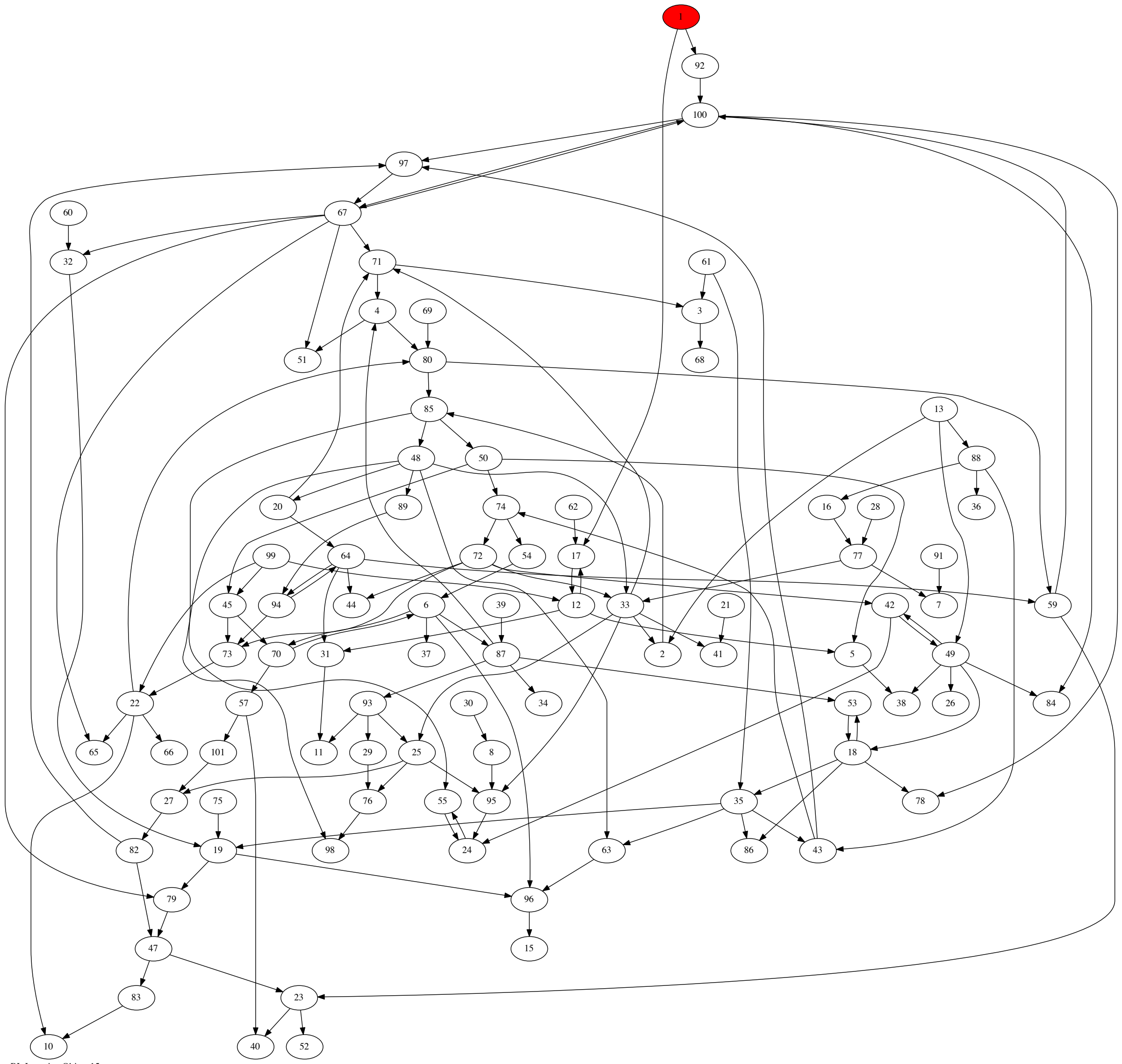


BL Looming Object 12

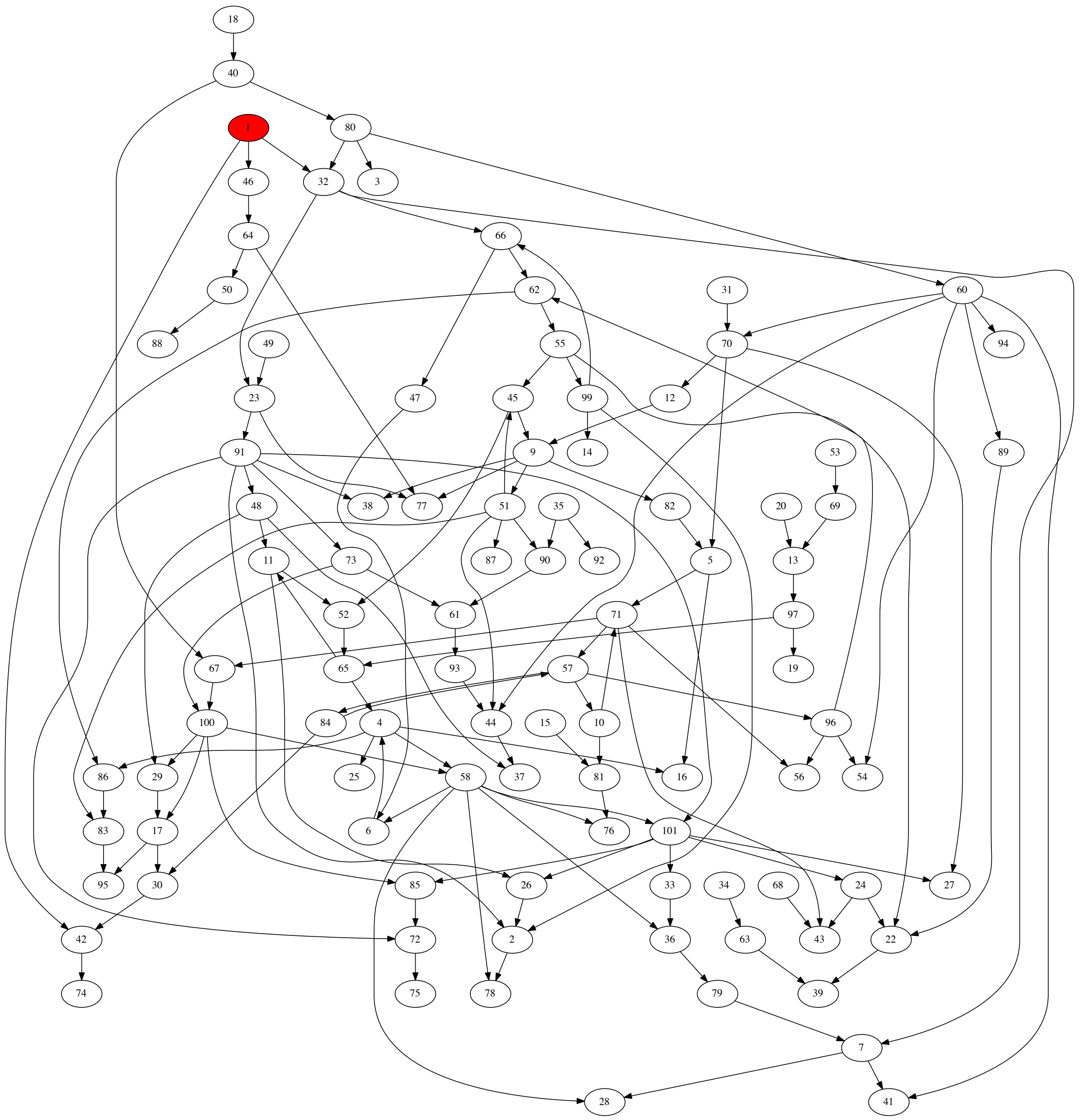

BL Looming Object 13



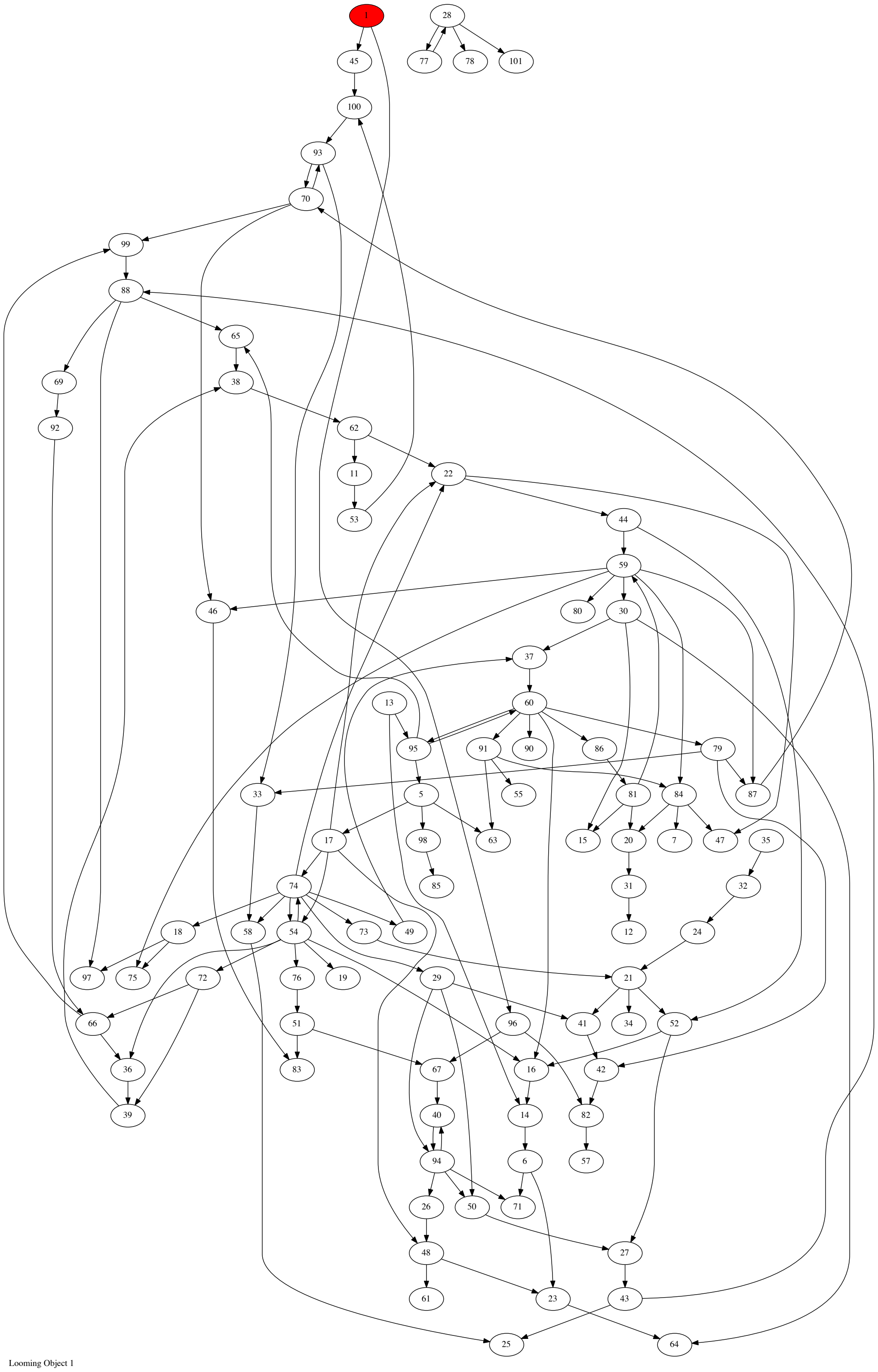
BL Looming Object 14



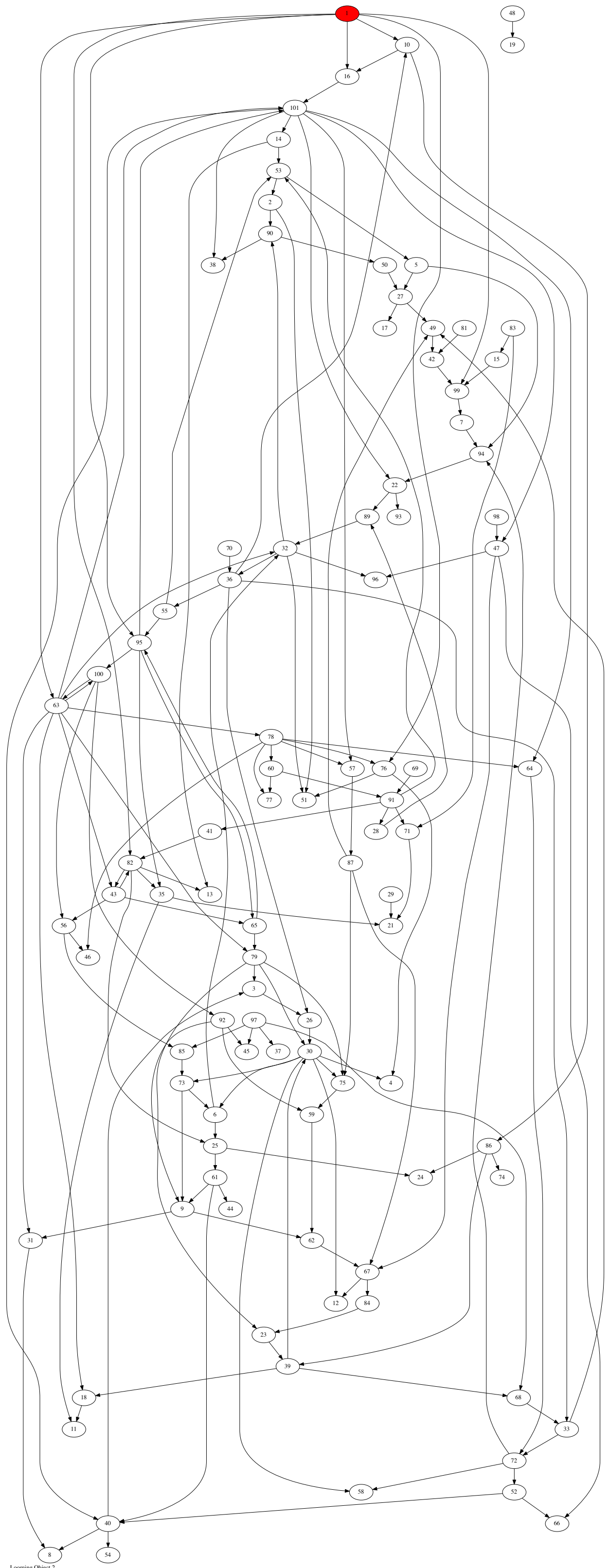
BL Looming Object 15

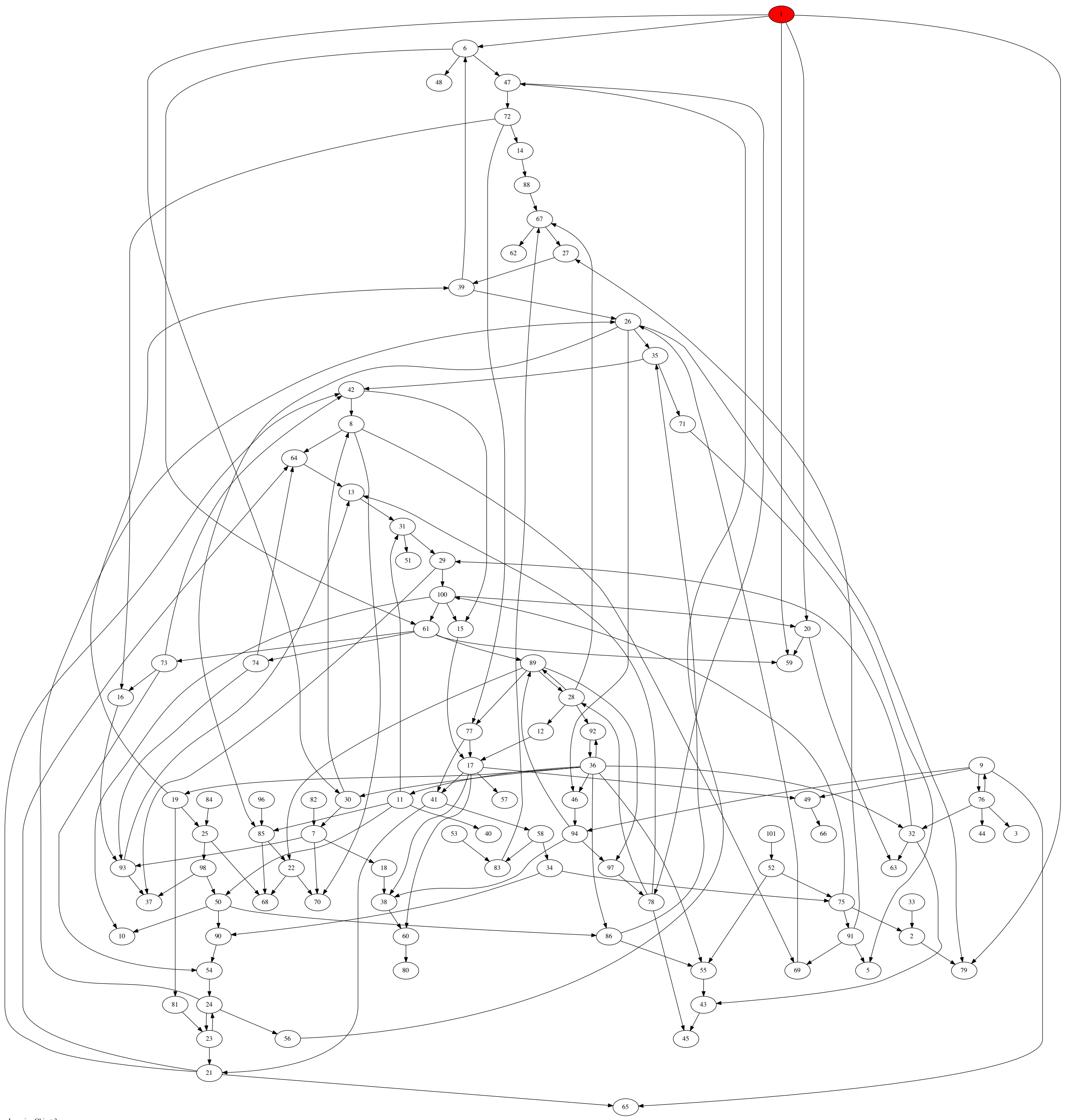


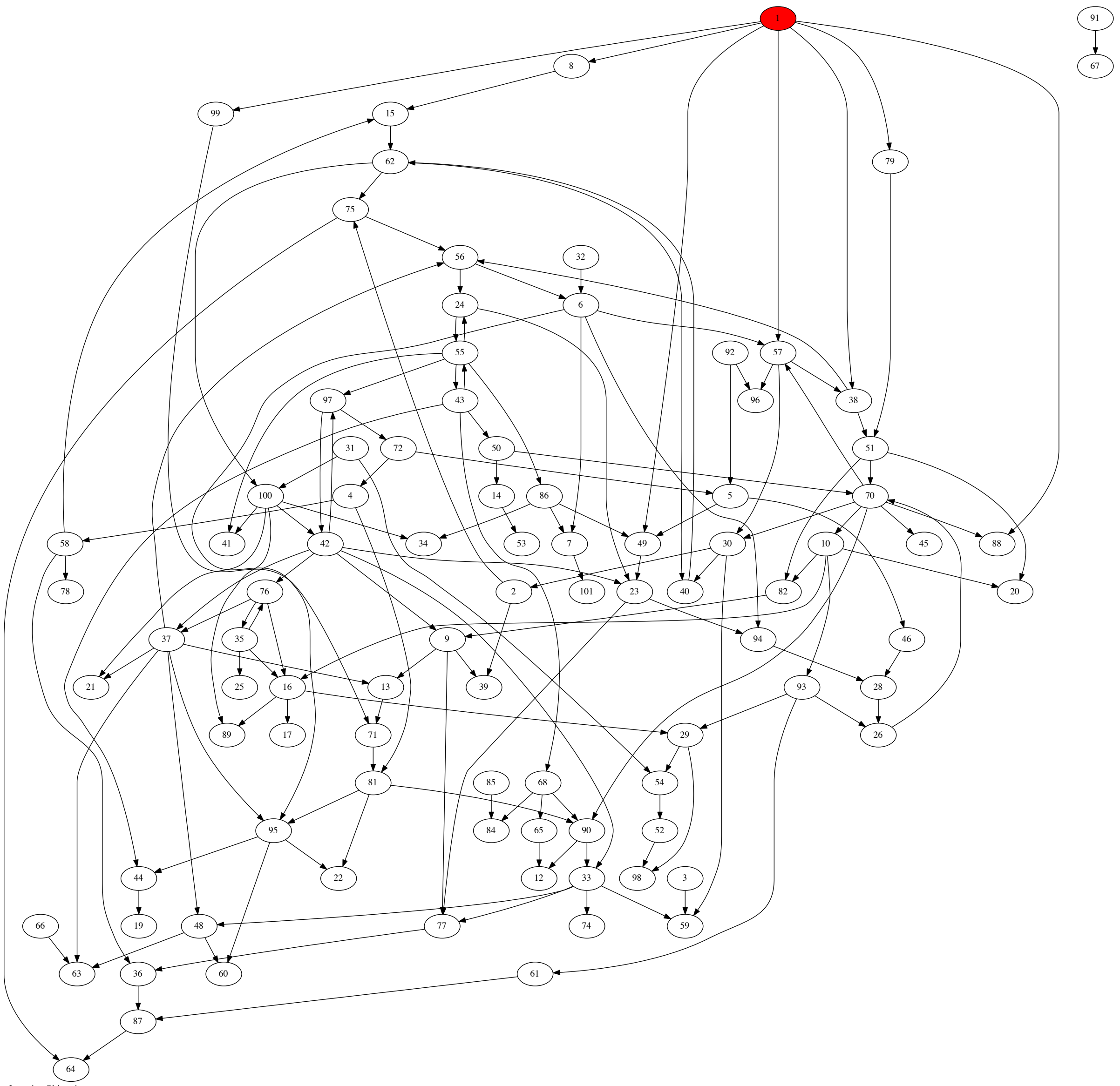
BL Looming Object 16

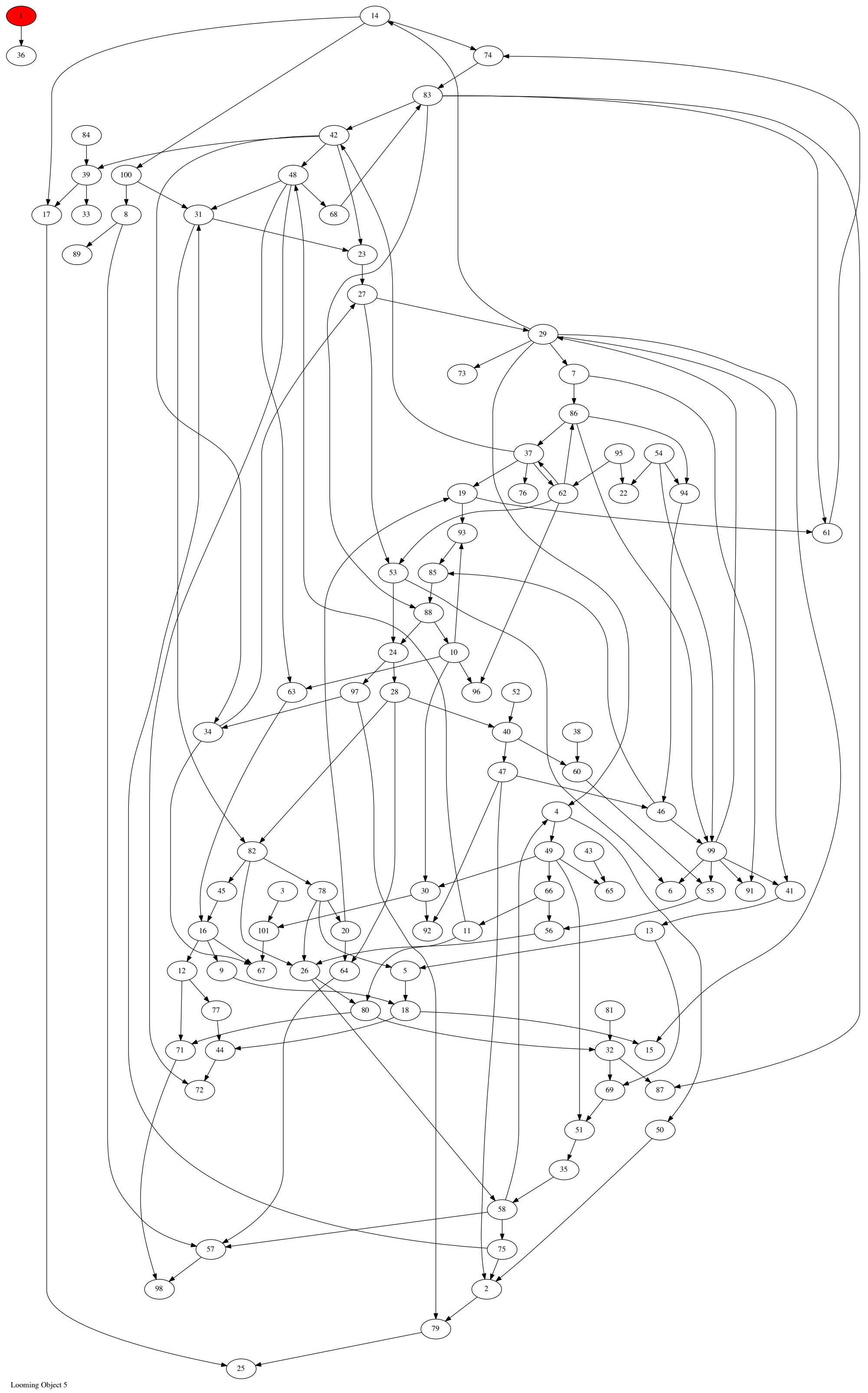


Looming Object 1

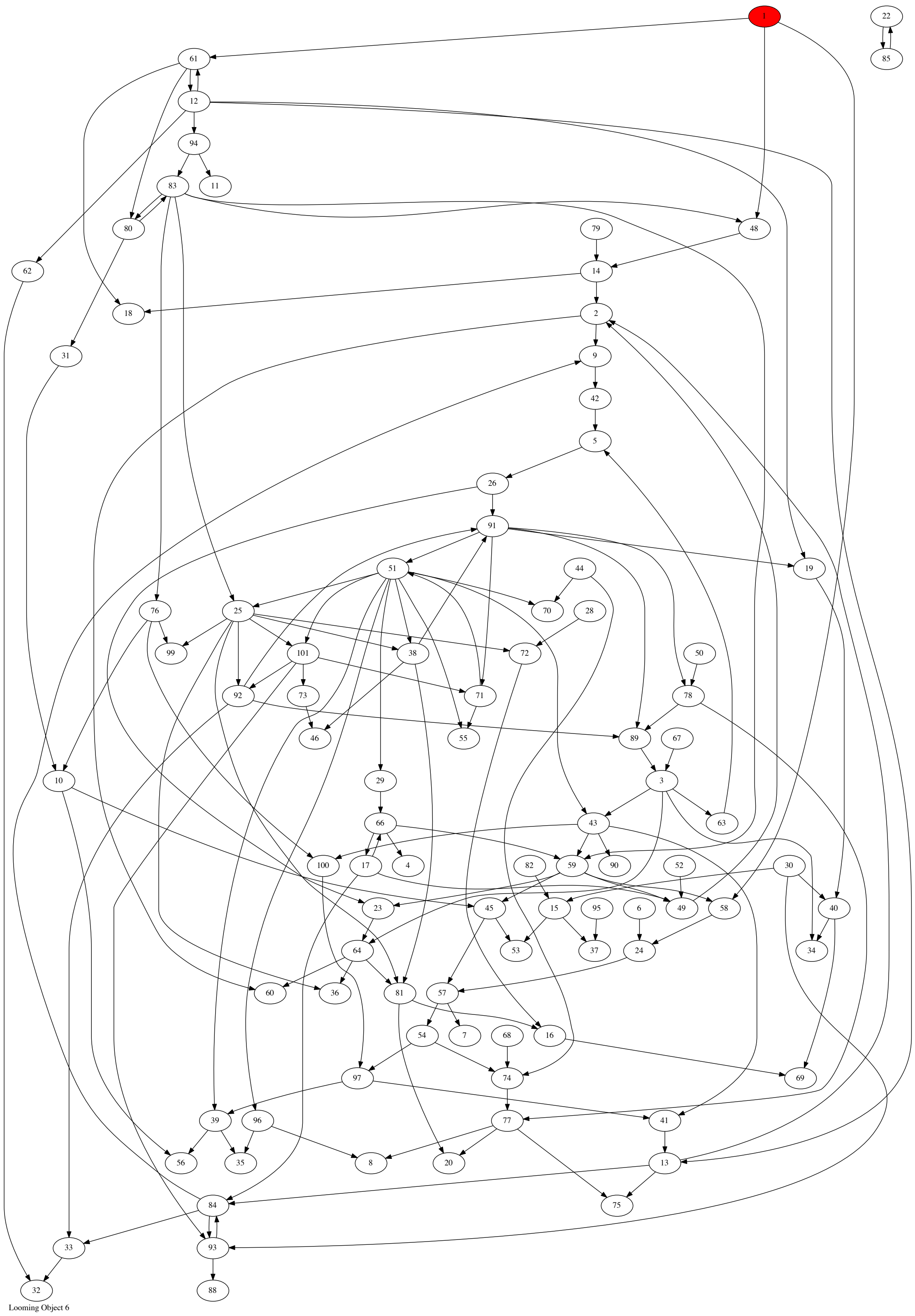

Looming Object 2


Looming Object 3

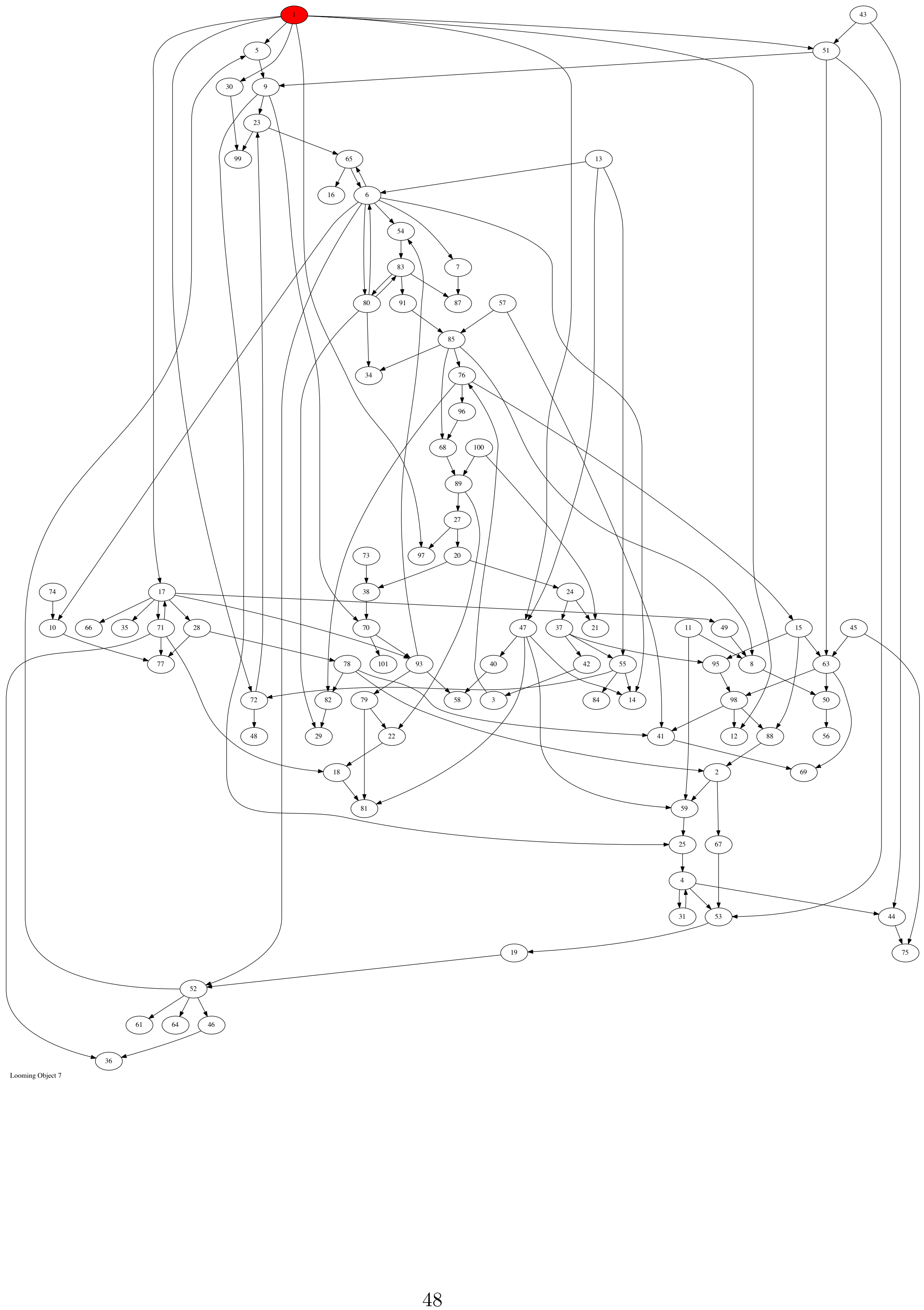

Looming Object 4

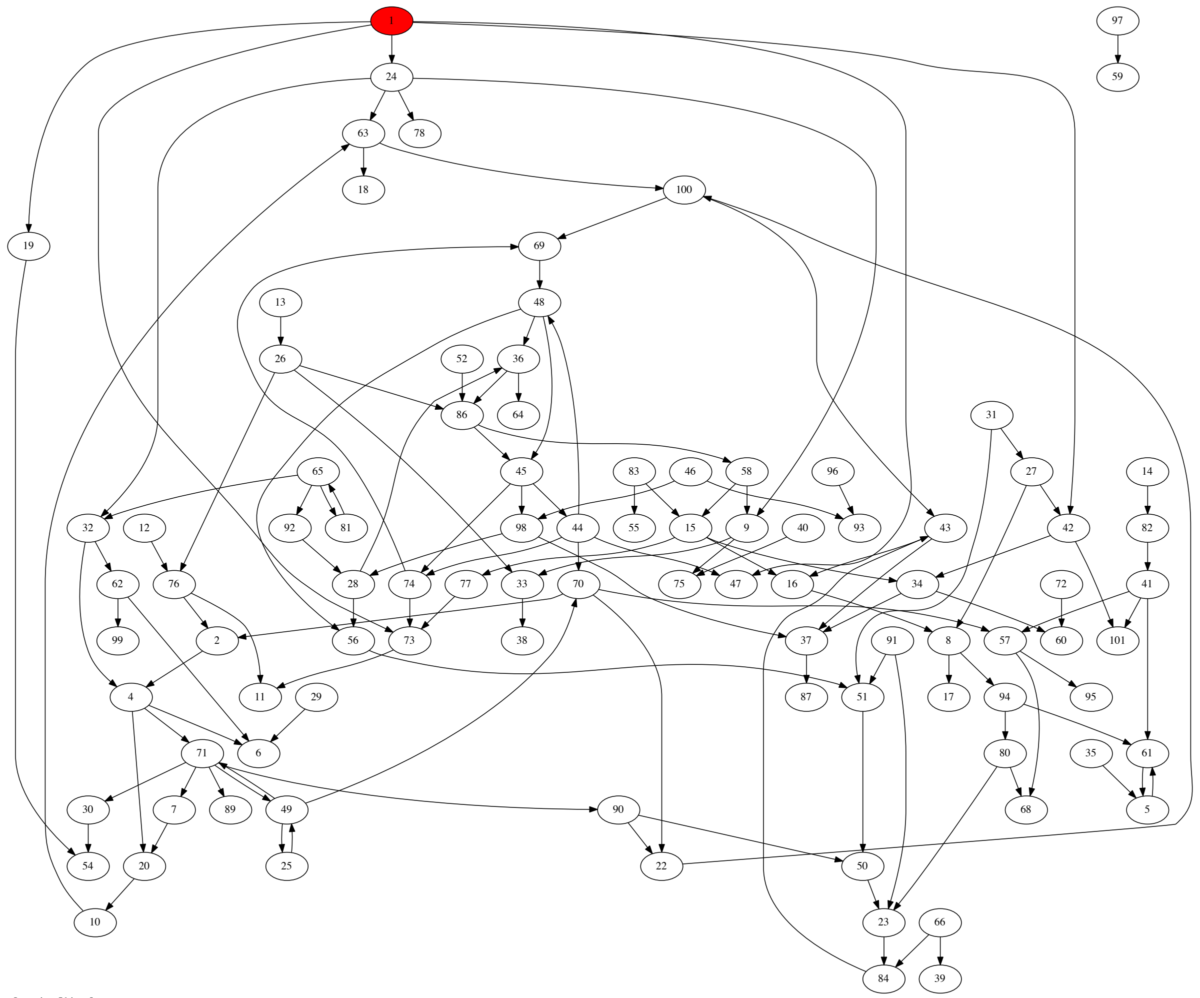


Looming Object 5

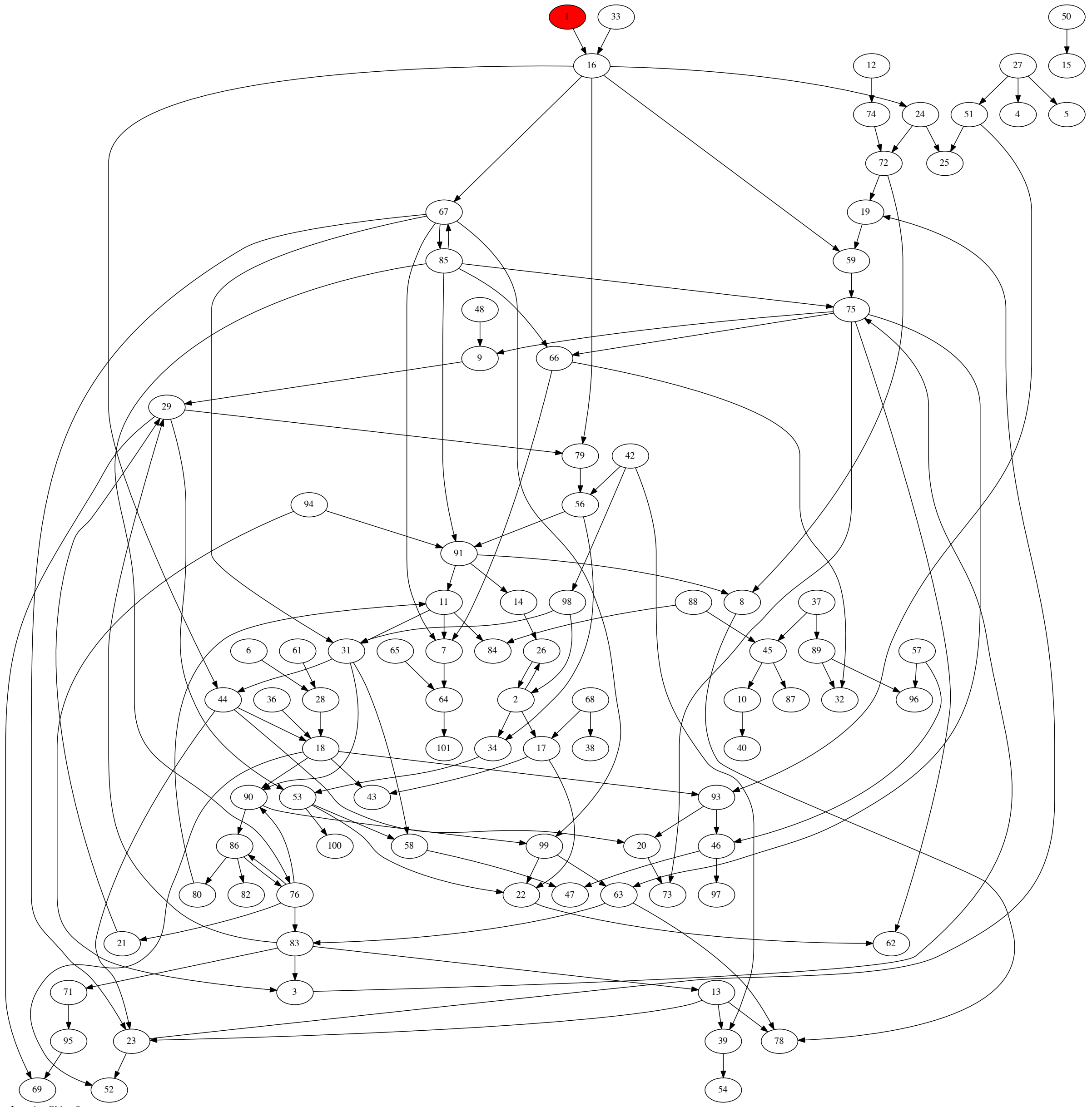


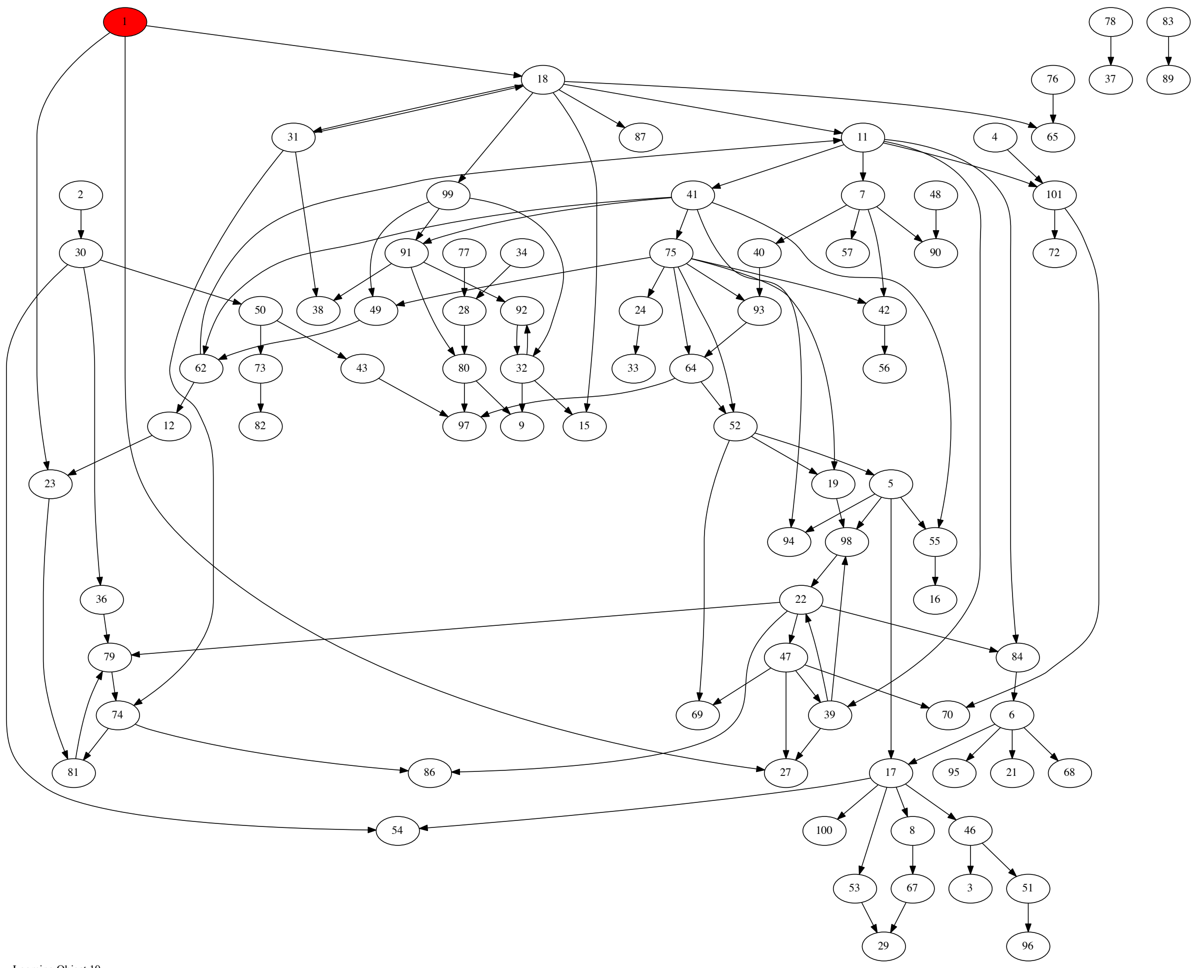
Looming Object 6


Looming Object 7

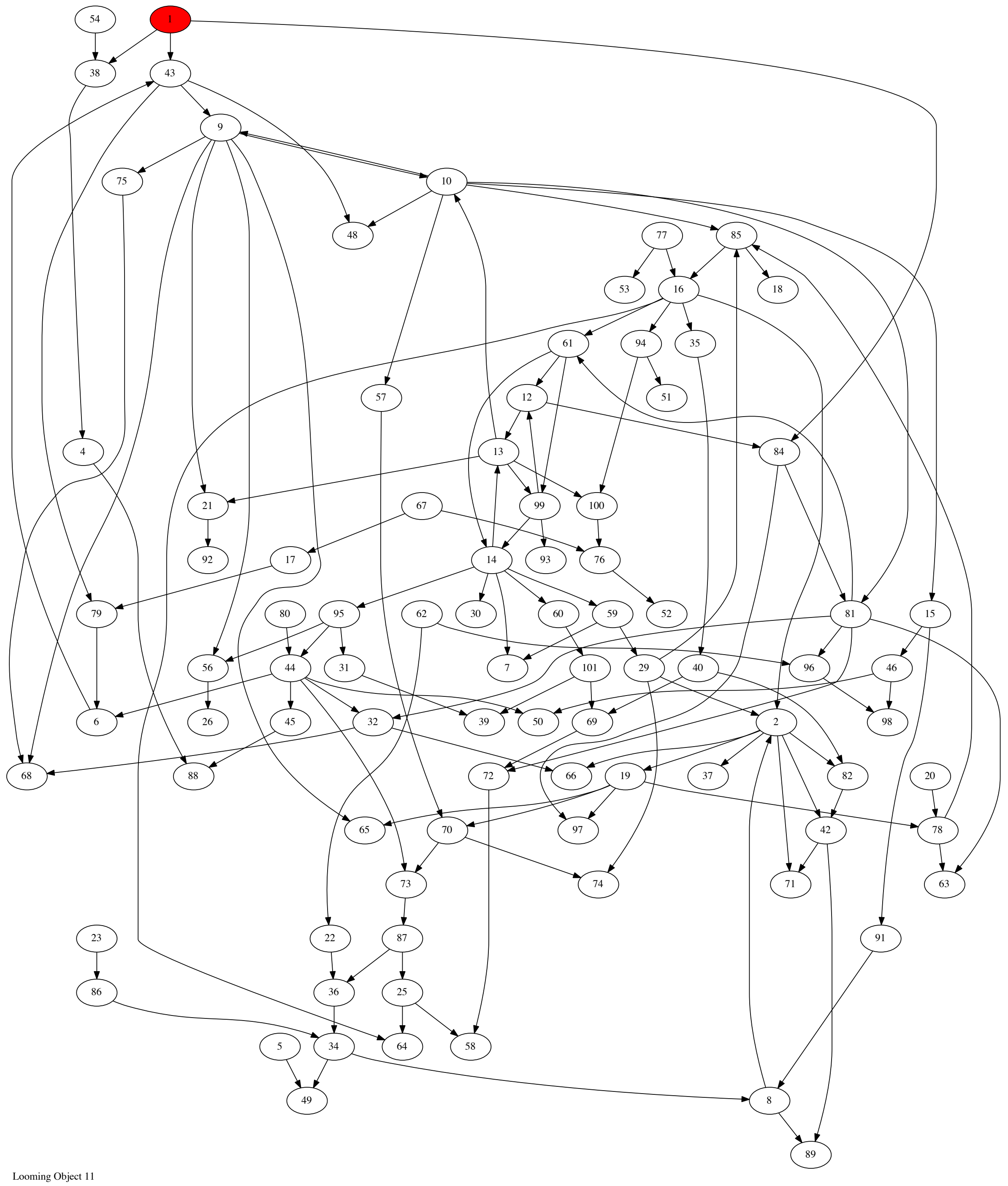


Looming Object 8

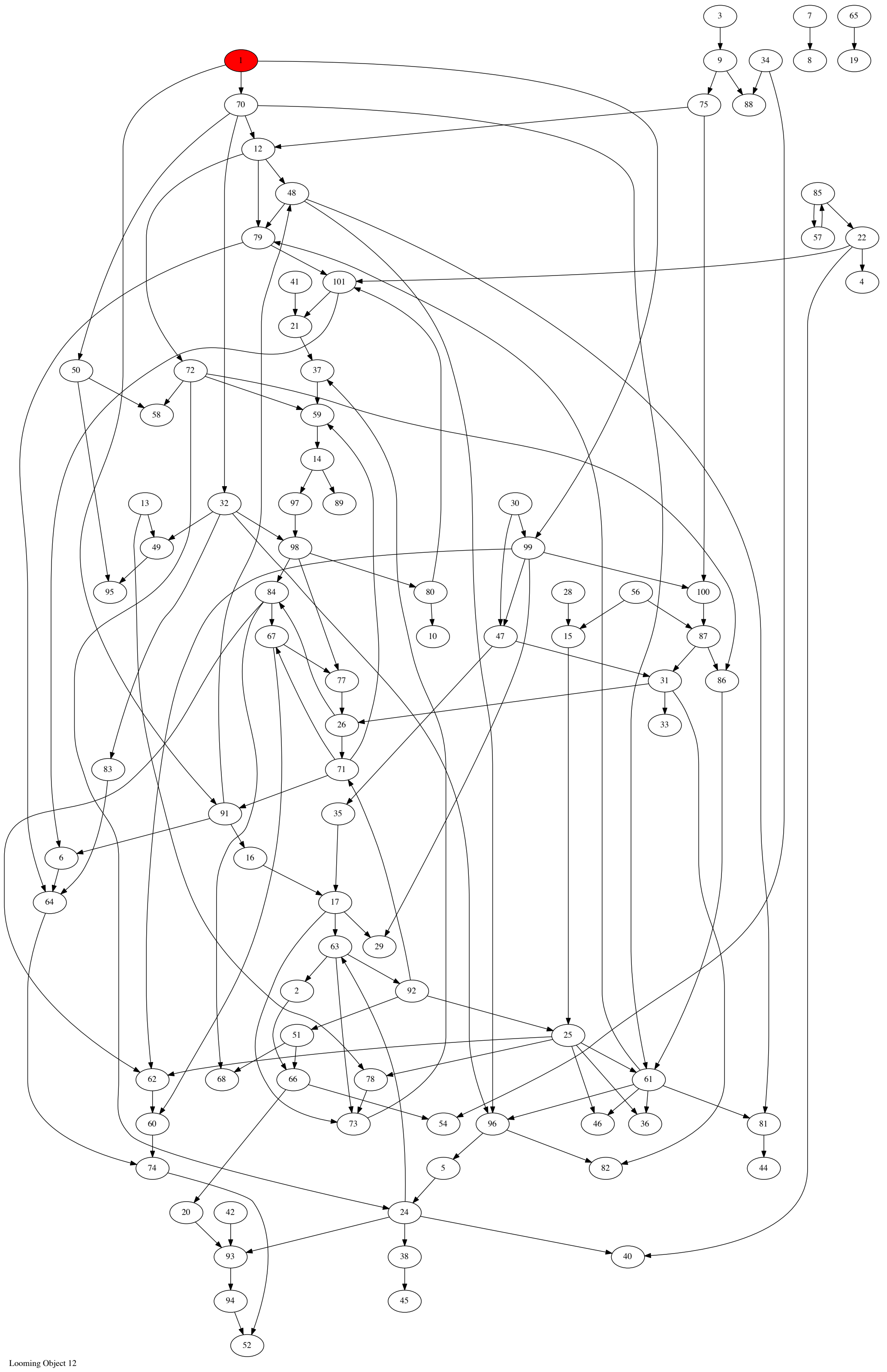

Looming Object 9



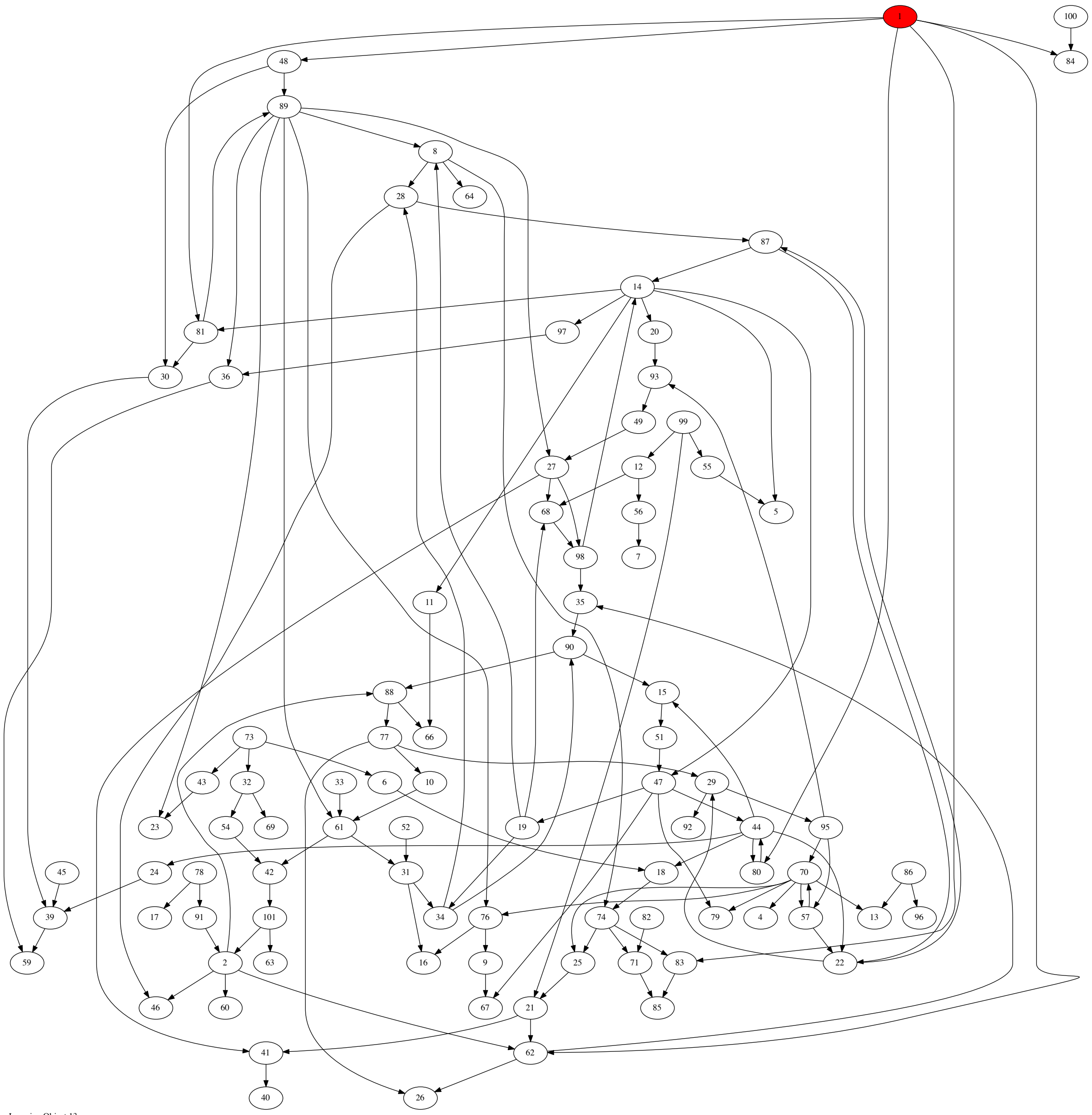
Looming Object 10



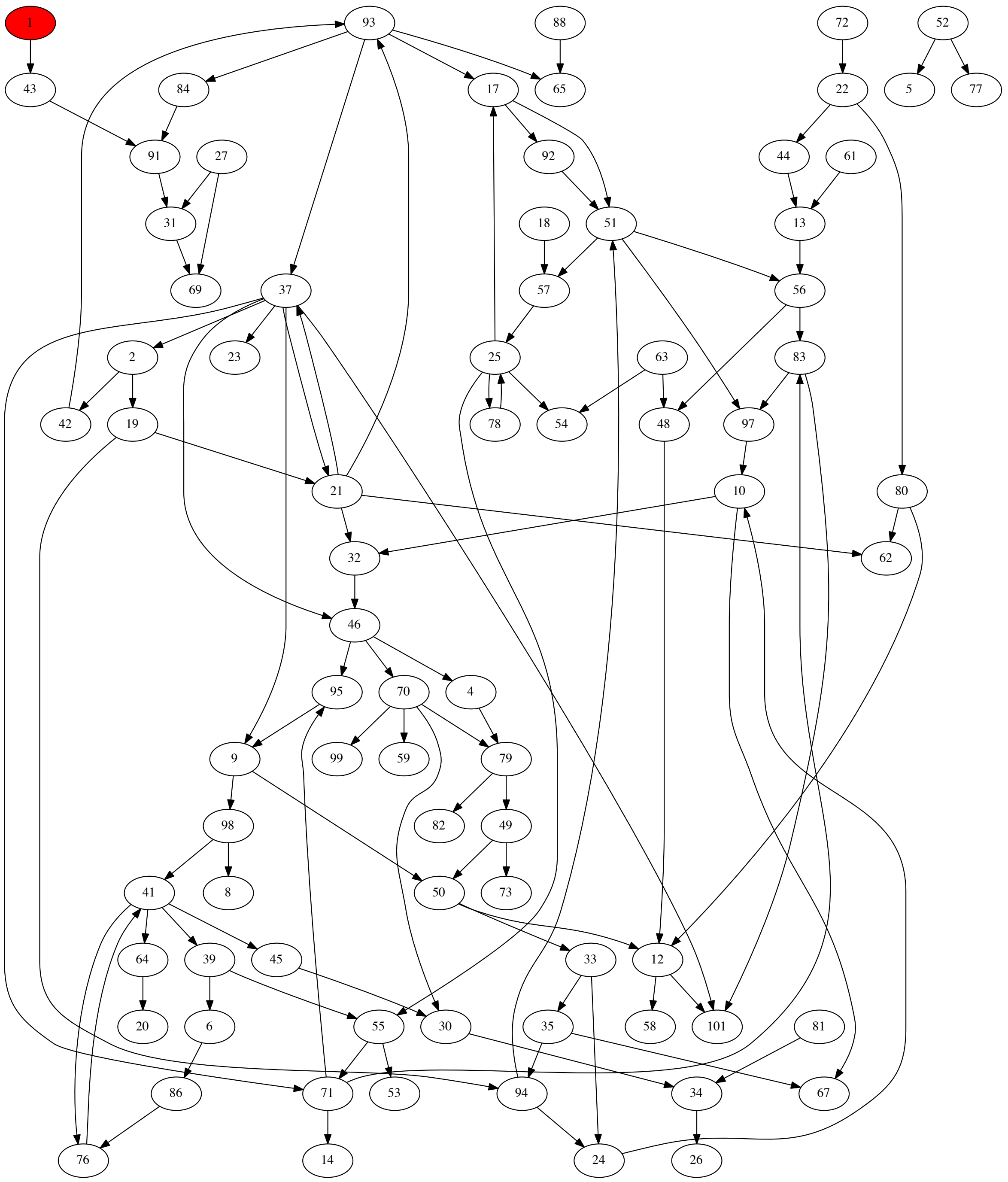
Looming Object 11



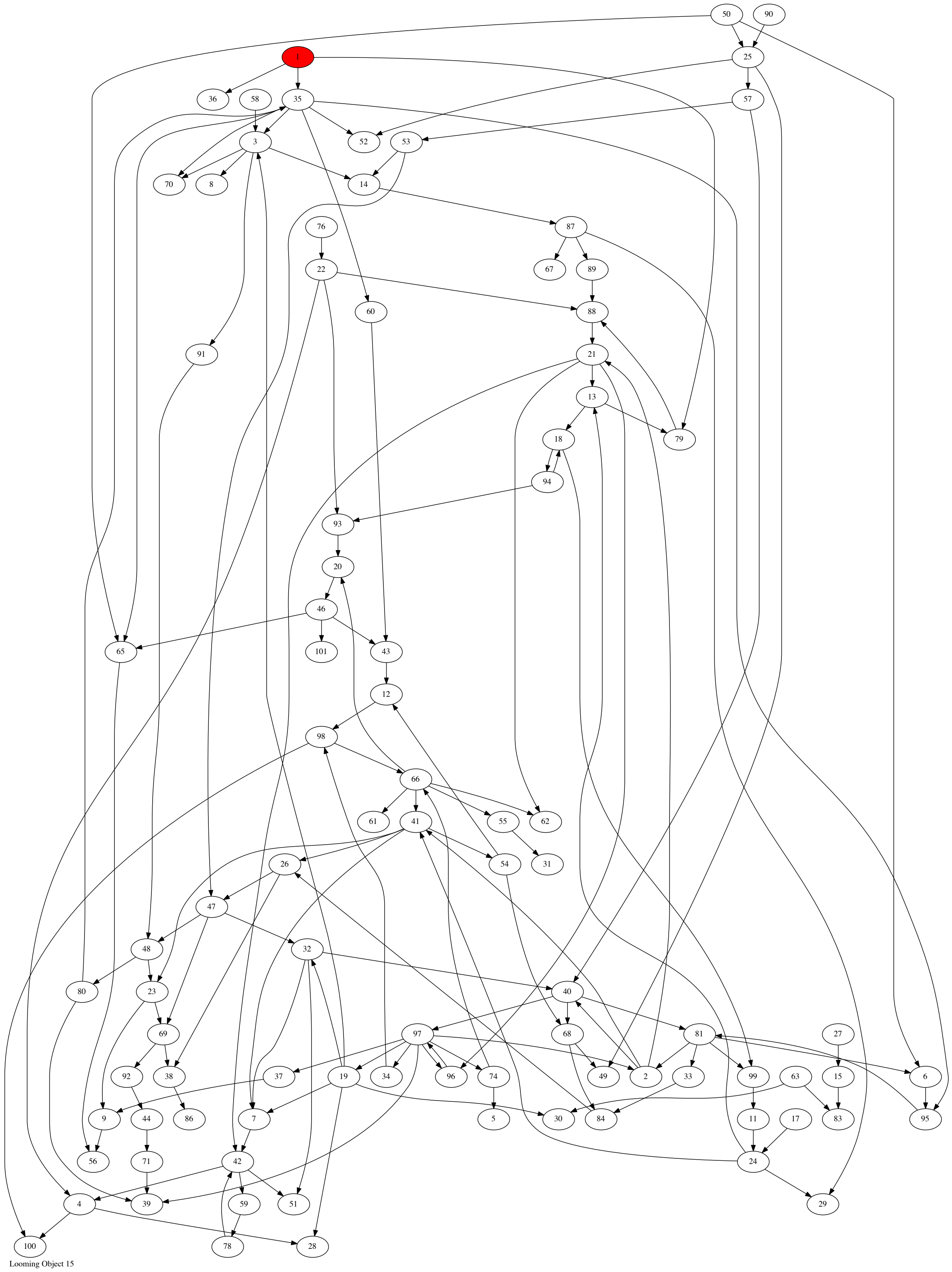
Looming Object 12

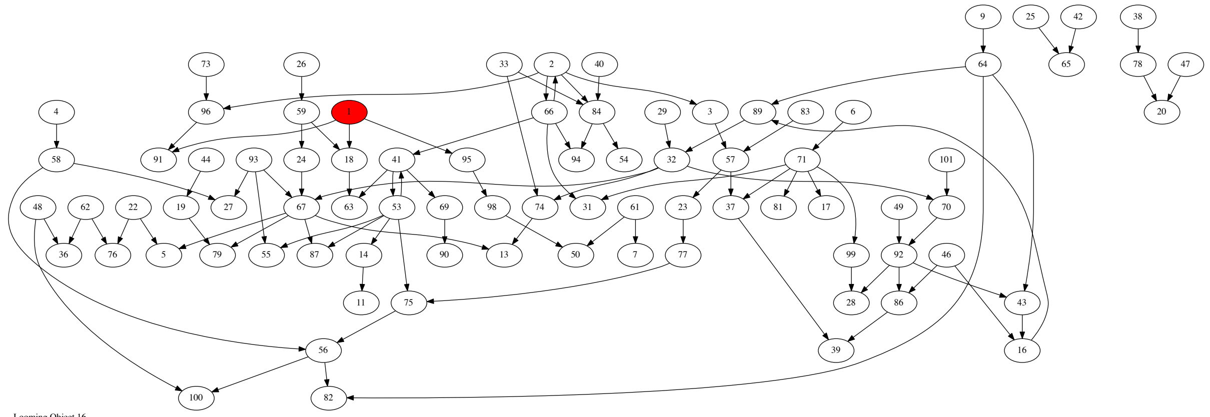


Looming Object 13



Looming Object 14


Looming Object 15



Looming Object 16