# Temporal Reverse Engineering

Danny Quist

Colin Ames

Blackhat USA 2008

# Danny Quist

- Co-founder Offensive Computing, LLC
- Ph.D. Candidate at New Mexico Tech
- Reverse Engineering Instructor
  Infosec Institute
- dquist@offensivecomputing.net

# Colin Ames

- Security Researcher, Offensive Computing
- Steganography Research
- Penetration Testing
- Reverse Engineering
- Malware Analysis
- amesc@offensivecomputing.net

# Overview of Talk

- Current Techniques
  - Where they work
  - Where they fail
- What is Temporal Reverse Engineering?
- Process pausing techniques
- Visualization Methods
- Applications and Demos

# Reverse Engineering

- RE is hard
- Goal: Figure out how program works in minimal amount of time
- Expensive (We don’t work cheap)
- Time consuming

# Dominant Strategies

- Static Analysis
  - IDA Pro, dumpbin
  - Figure out program flow
  - Search for strings
  - API Call graphing

# Dominant Strategies

- Dynamic Analysis
  - – Watch for changes on the system
    - Registry, files, network
  - – Monitor System calls
  - – Tools more accessible to unskilled people
  - – Sysinternals, Winanalysis, etc.

# Pros

**Static Analysis**

- Details
- Precision, full code reversal possible
- Good tools available
- Lots of source level static analysis programs
- Antivirus
  - It’s profitable

**Dynamic Analysis**

- Fast
- Lower barrier to entry
- High level overview
- Good tools
  - Sysinternals
  - Winanalysis
  - CWSandbox

# Cons

**Static Analysis**

- Too much detail
- Full code reversing not necessary
- Tools cumbersome, take awhile to learn
- Source level analysis full of false positives
- Antivirus
  - Doesn’t scale

**Dynamic Analysis**

- Misses details
- Encourages “next->next->next” analysis
- Tools easily subverted

# Bridging the Gap

- Fundamental problem:
  - Know ***when*** to analyze, not what
  - Data changes, need to track and respond to those changes
- Techniques
  - Debuggers
  - Pagefault assisted debugging (Saffron)
  - Dynamic Translation
  - Sandboxing

# Monitoring Program Execution

- Intel PIN
  - Dynamic instrumentation library
  - Extensible
  - Awesome API
- Saffron
  - Covert monitoring
  - Limited back tracking

# Visualization

- Monitor program execution with visualization techniques
- Valuable insight into process monitoring
- Integration with IDA and Olly

# What about program flow tracing?

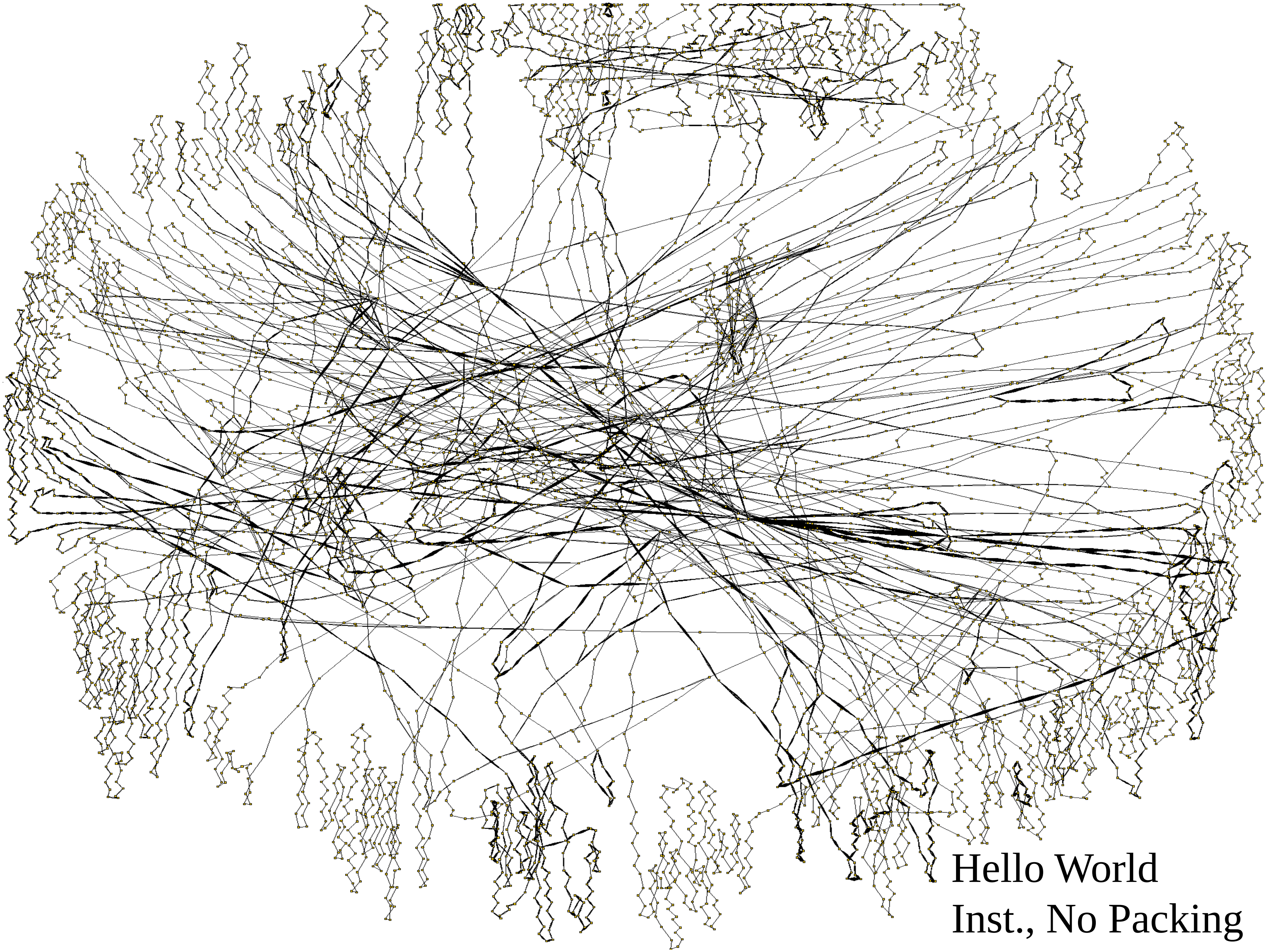
- Visualization should be able to answer a question quickly
- How can we apply this to reverse engineering?
- Find a way to quickly represent information

# Find the Unpacking Loops

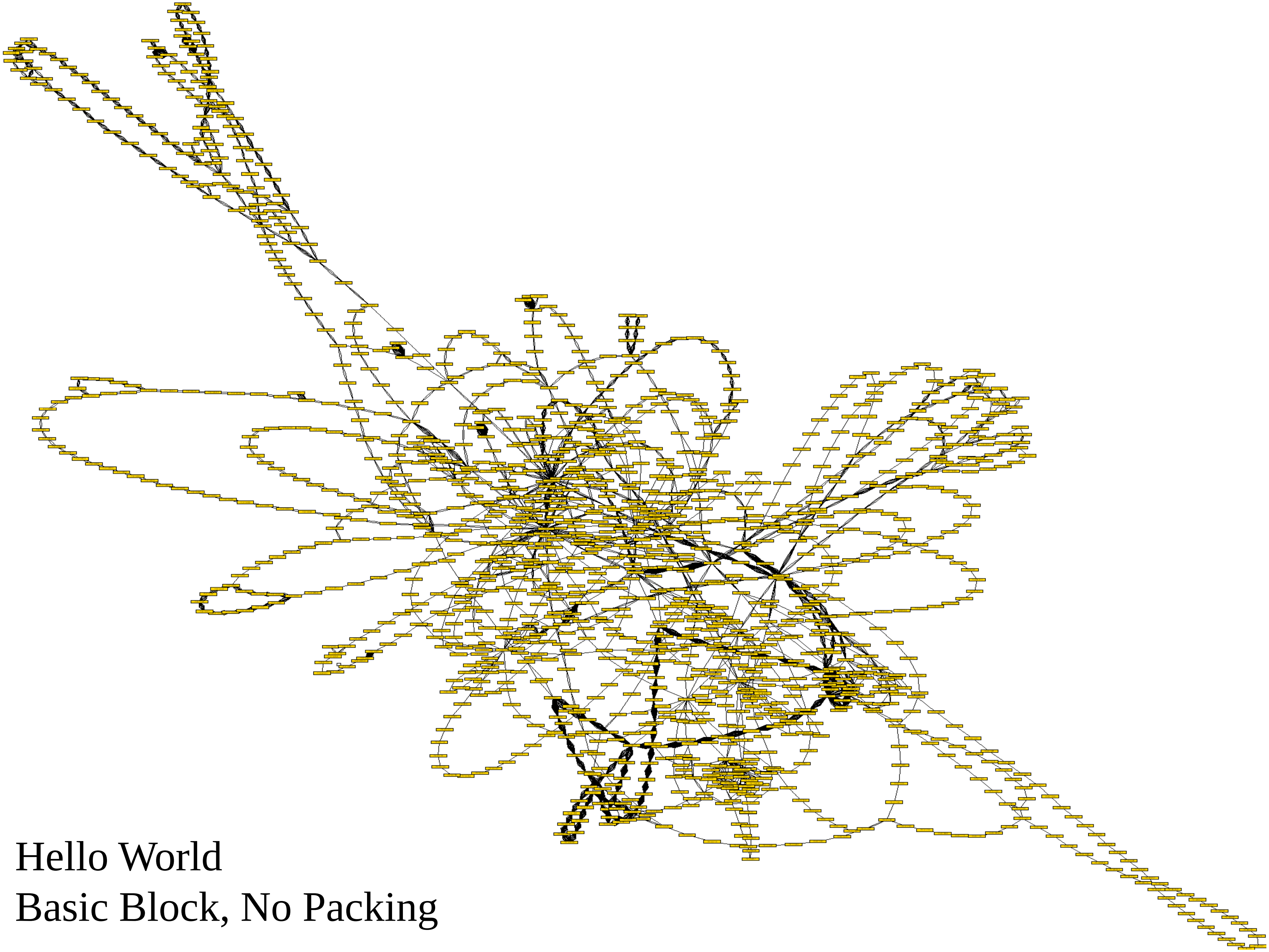
- Simple hello world program

```
int main(int argc, char **argv)
{
    printf(“Hello, world\n”);
    return 0;
}
```

- Packers used
  - ASPack, FSG, PECompact, UPX



Hello World
Inst., No Packing

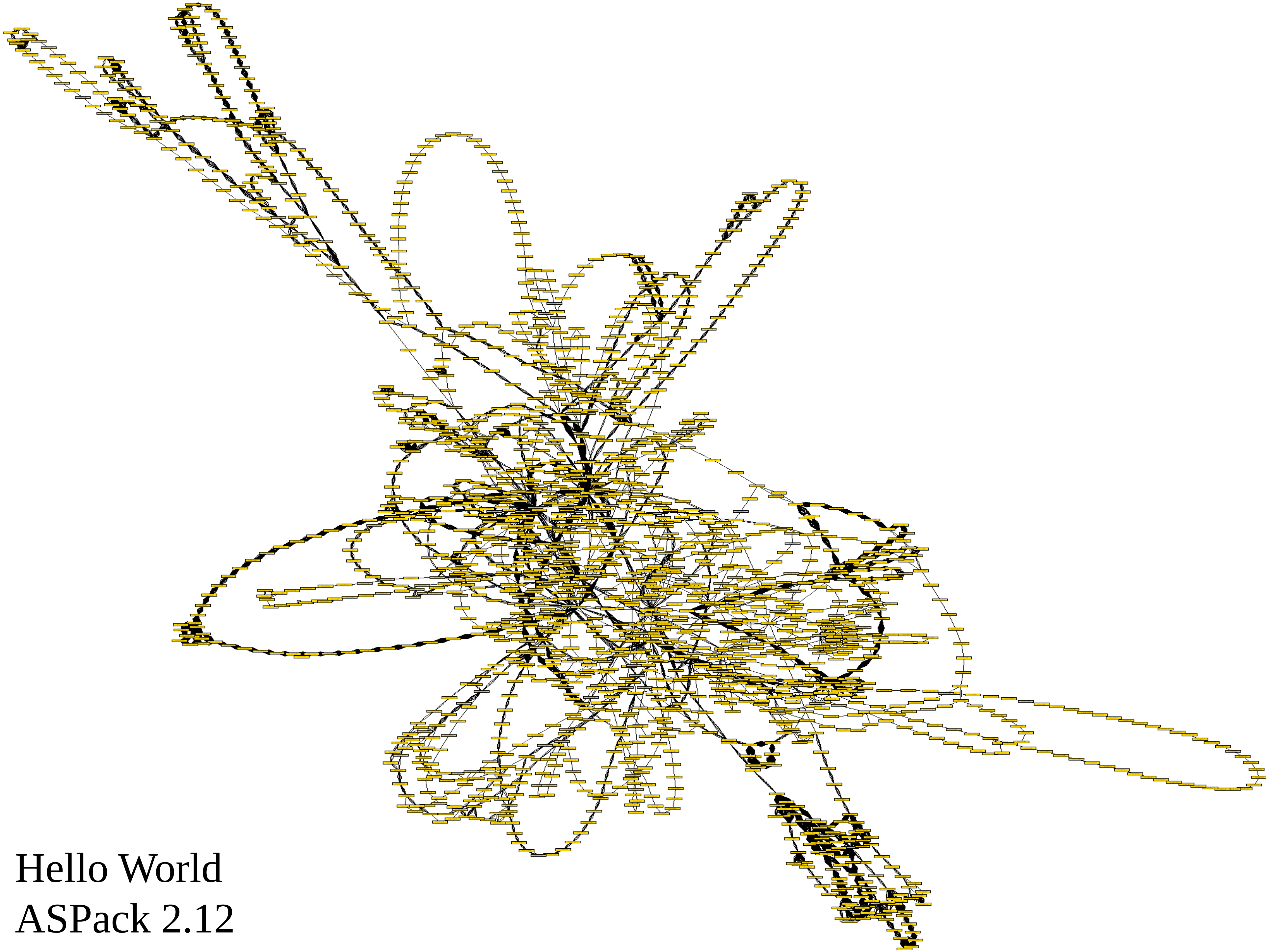


Hello World

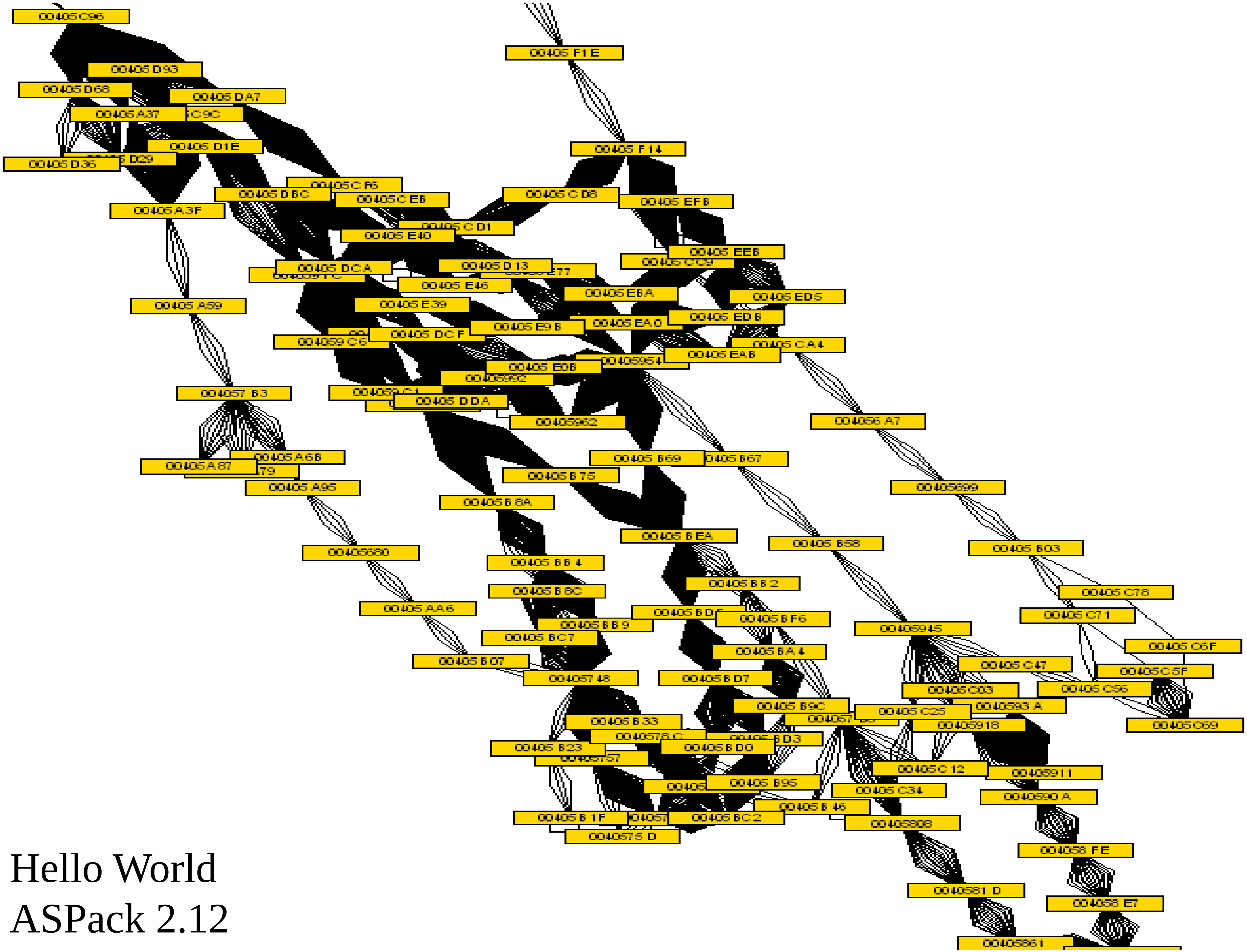
Basic Block, No Packing

# Adding Packers

- Should be able to find the following:
  - Packing loop
  - Main program
- Minimize extraneous information
- Reducing analyst time is the key

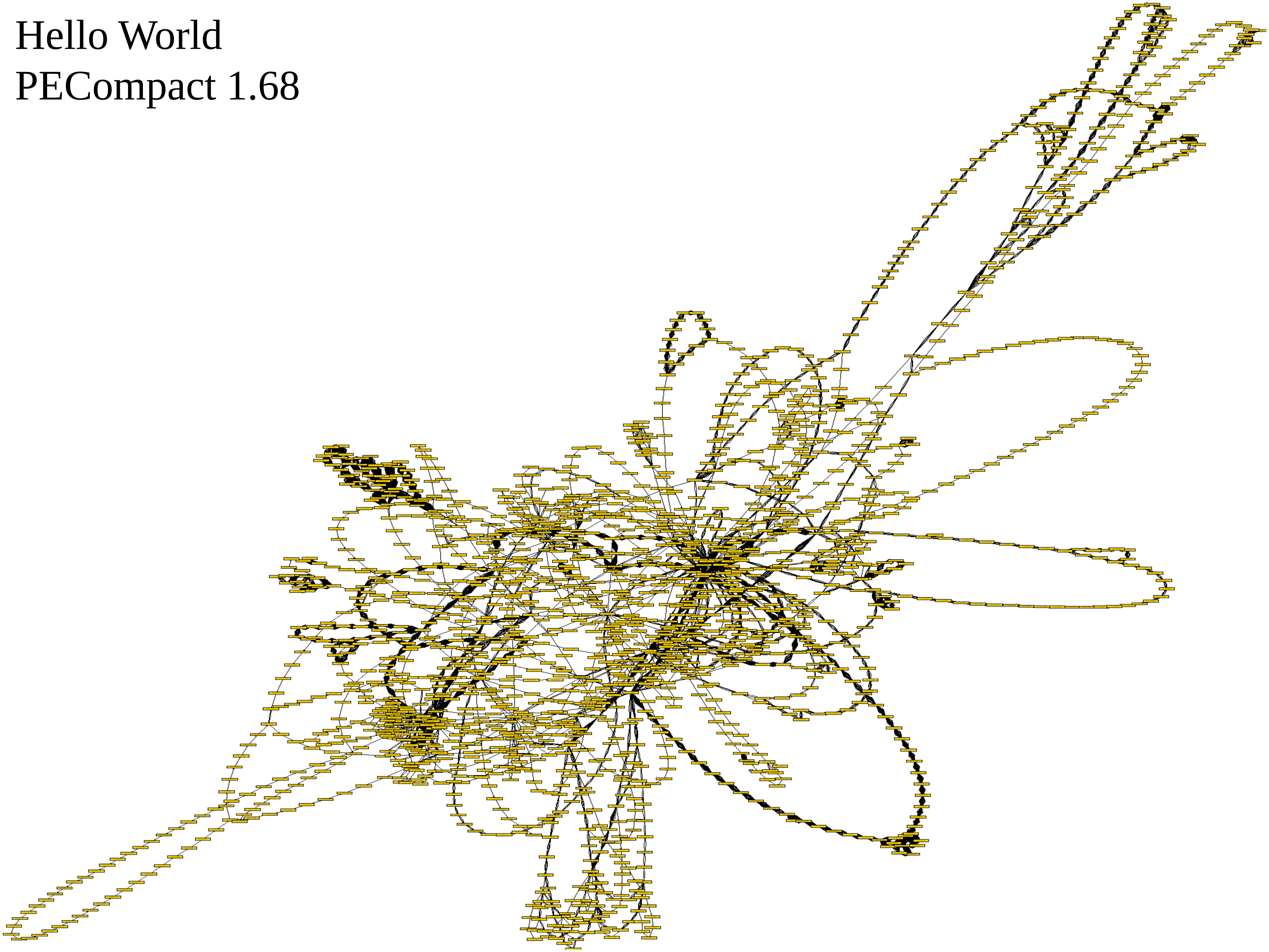


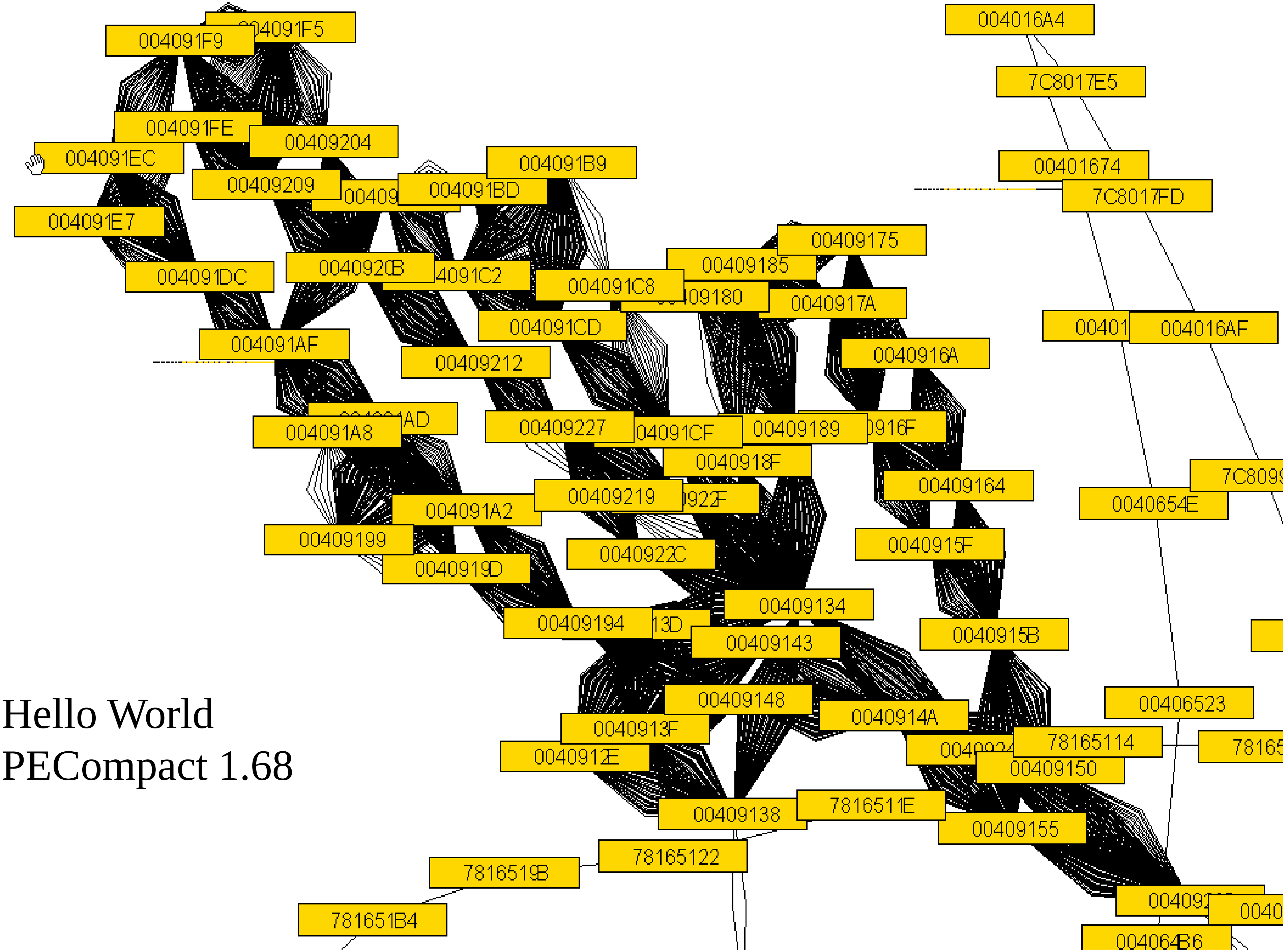
Hello World
ASPack 2.12



Hello World
ASPack 2.12

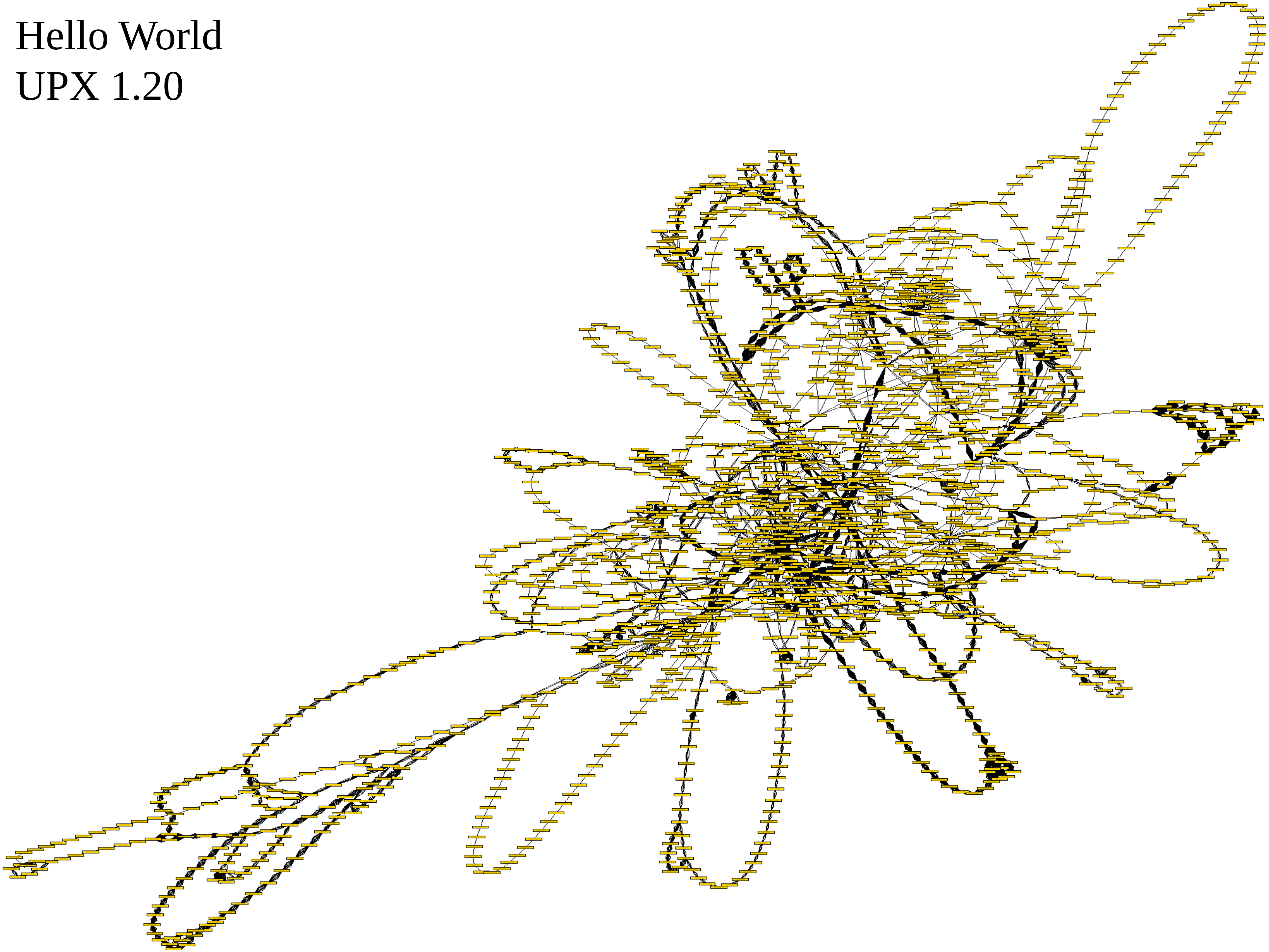
Hello World
PECompact 1.68

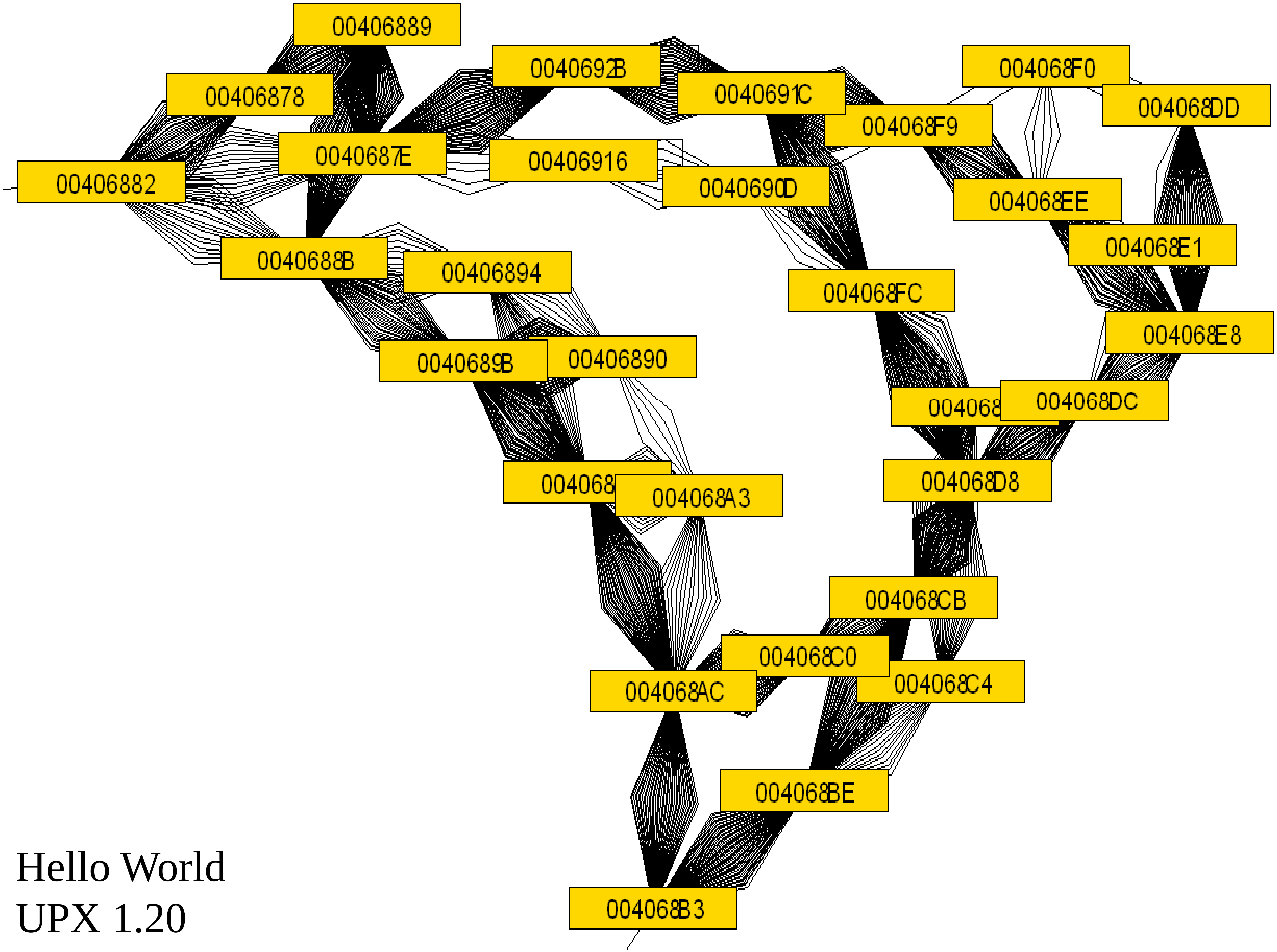




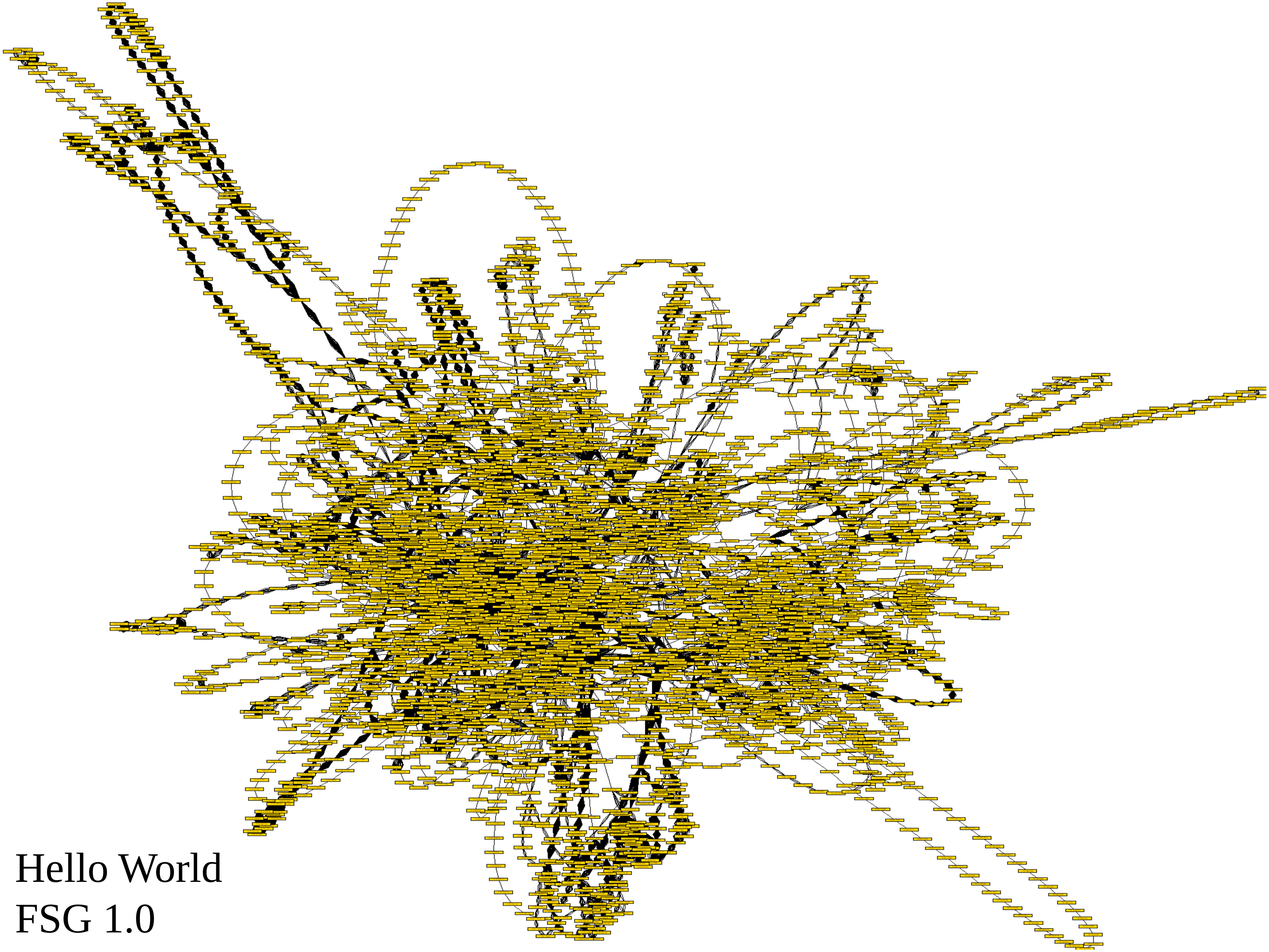
Hello World
PECompact 1.68

Hello World
UPX 1.20

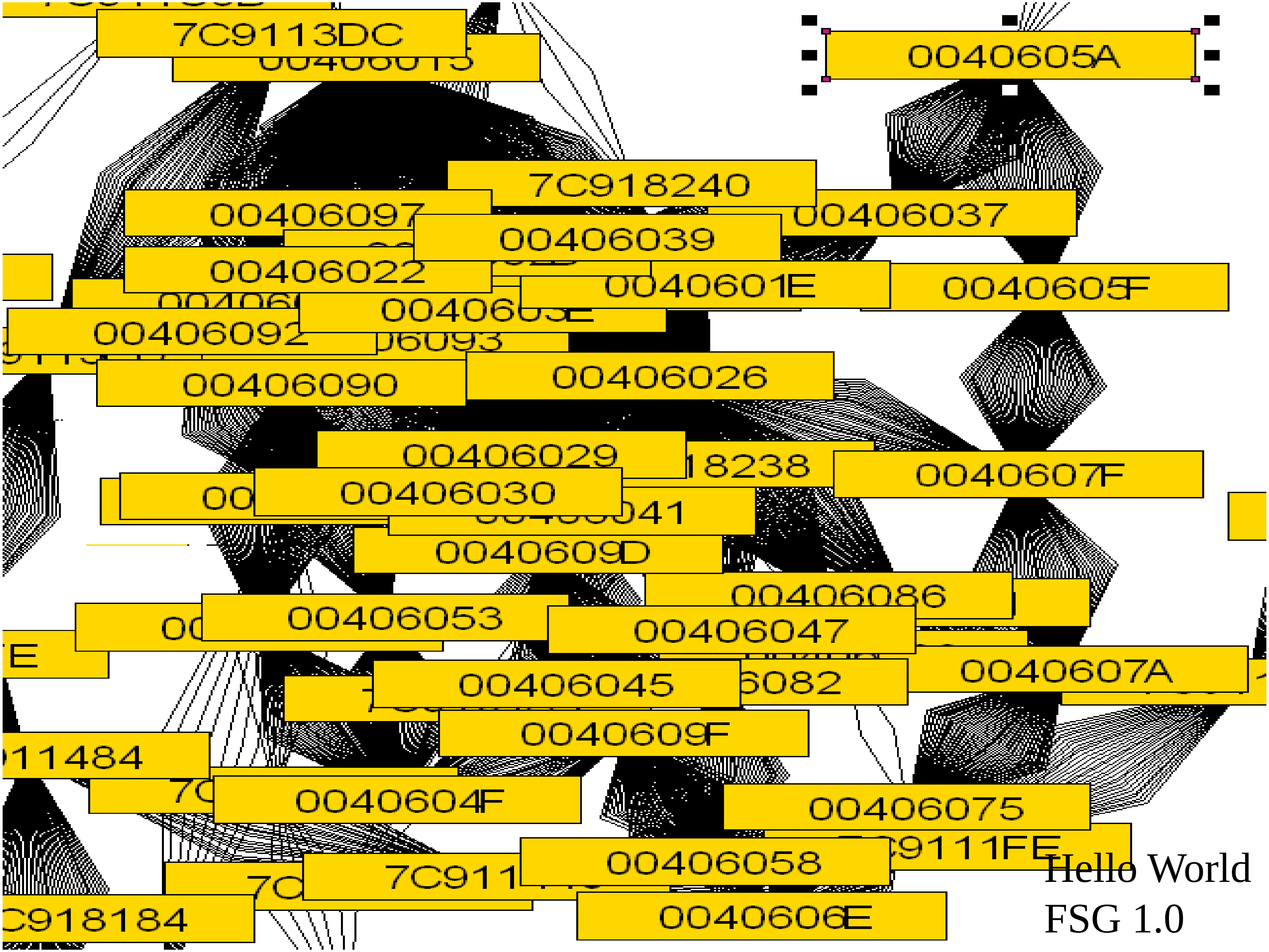




Hello World
UPX 1.20



Hello World
FSG 1.0



Hello World
FSG 1.0

# Temporal Control of Execution

- Previous methods
  - Virtual machines
  - Debuggers
  - Simple restart
- Problems
  - Time intensive
  - Algorithmic analysis does not need full system restore

# Snapshotting

- Determine when to snapshot
  - Instruction
  - Basic block
  - Page access

# Snapshotting

- Preservation of state
  - Register contents
  - Stack contents
  - CPU State
  - Memory

# Existing Snapshot Tools

- OS Suspend
- Cryopid
- Memory Paging
- OS Scheduler

# Isolating Important Data

- Memory maps
- Memory hotspots
- Colometric memory visualization
- Data motion with silhouette hulls

# Rebuilding PE files for IDA

How IDA creates its import section .idata and populates subviews Imports, Names

- – IMAGE_DIRECTORY_ENTRY_IMPORT
  - • RVA (Relative Virtual Address) to Import Directory
- – IMAGE_IMPORT_DESCRIPTOR’s
  - • OriginalFirstThunk
    - – RVA to INT (Import Names Table)
  - • FirstThunk
    - – RVA to IAT (Import Address Table)
- – Scan’s Code for call’s in INT
  - • Prepends internal functions to .idata section

# Rebuilding PE files for IDA

## Recovering INT from packed or encrypted PE

- Unpack using Saffron
  - Discover OEP
- Enumerate Loaded Modules
  - CreateToolhelp32Snapshot, Module32First
- Scan Process heaps for Module Address
  - Translate Virtual Address into RVA
- Rebuild INT and IAT
  - Dump Process memory

# Malware Demo

# Information Protection Demo

# Conclusion

- Quick way to check memory changes
- Shortens analyst time
- Integrate with existing apps
- Visualization adds clarity

# References

- Visualization Grand Challenges: Illuminating the Path http://nvac.pnl.gov/docs/RD_Agenda_NVAC_chapter1.pdf
- Dynamic Data Visualization of Meteorological Data ASA-JSM Data Exposition, 2006
- Visual Signatures in Video Visualization IEEE Transactions on Visualization and Computer Graphics, Vol.12, No. 5, September/October 2006
- Static Visualization of Dynamic Data Flow Visual Program Execution Proceedings of the Sixth International Conference on Information Visualization, IV 2002
- Hoglound, G., McGraw, G., Exploiting Software: How to Break Code, *Chapter 3, Addison Wesley, 2004*
- Amini, P., Process Stalker, *OpenRCE, http://pedram.redhive.com/code/process_stalker/*
- Amini, P., PaiMei, *OpenRCE http://www.openrce.org/downloads/details/208/PaiMei*
- Eagle, C., x86emu, *http://ida-x86emu.sourceforge.net/*
- P. Ferrie, Attacks on Virtual Machines, Symantec Advanced Threat Research, 2007
- C. Luck, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V.J. Reddi, K. Hazelwood, Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation, *Proceedings of the 2005 Conference on Programming and Language Design and Implementation*, 2005
- Oreas GDE, *http://www.oreas.com/index_en.php*

**Latest slides and code can be found on offensivecomputing.net**