Tiptoe: A Compositional Real-Time Operating System

Christoph Kirsch
Universität Salzburg

ARTIST2 Workshop on Foundations and Applications of Component-Based Design, Salzburg, September 2007

tiptoe.cs.uni-salzburg.at

- Silviu Craciunas* (Benchmarking)
- Hannes Payer (Memory Management)
- Ana Sokolova* (Theoretical Foundation)
- Horst Stadler (I/O Subsystem)
- Robert Staudinger* (Kernel)

*Supported by Austrian Science Fund Project P18913-N15

Process A || Process B

Operating System

Memory CPU I/O

"Theorem"

- (Compositionality) The time and space a software process needs to execute is determined by the process, not the system and not other software processes.
- (Predictability) The system can tell how much time and space is available without looking at any existing software processes.

“Corollary”

- (Memory) The time a software process takes to allocate and free a memory object is determined by the size of the object.
- (I/O) The time a software process takes to read input data and write output data is determined by the size of the data.

“Reality”

- A software process determines functional and non-functional behavior, for example:
  - 1ms/100ms CPU time (≠ 10ms/s)
  - 4MB/2s memory allocation rate
  - 1KB/10ms network bandwidth
  - 10J/100ms energy consumption

Outline

1. Memory Management
2. Concurrency Management
3. I/O Management

Toe A || Toe B

Tip

Memory CPU I/O

Outline

1. Memory Management
2. Concurrency Management
3. I/O Management

Tiptoe System

P2P Ethernet Connection

OR

Serial Connection

I/O Host Computer

Network

Disk

AD/DA

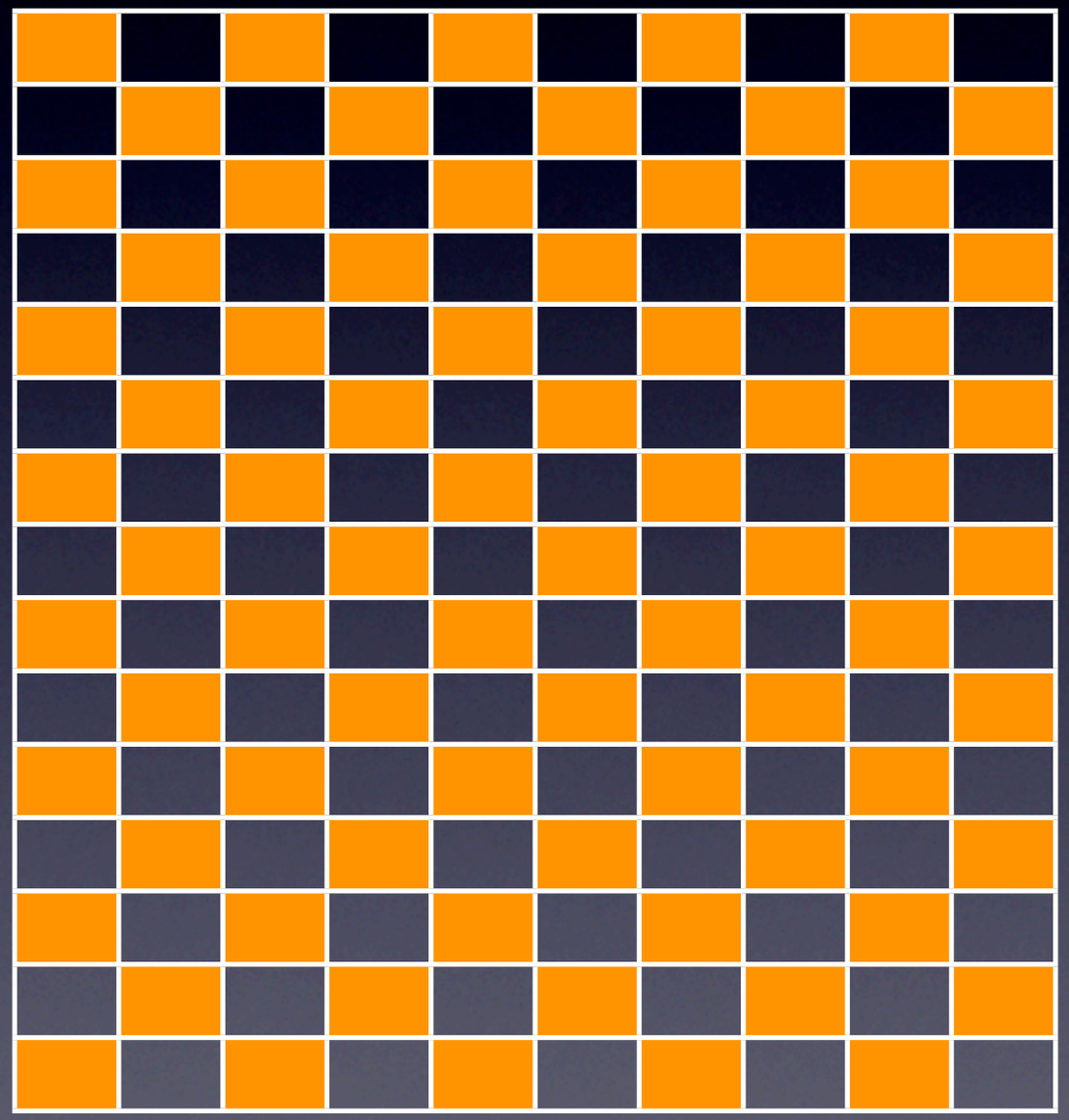
Outline

1. Memory Management
2. Concurrency Management
3. I/O Management

Goals

- malloc(n) takes at most TIME(n)
- free(n) takes at most TIME(n)
- access takes small constant time
- small and predictable memory fragmentation bound

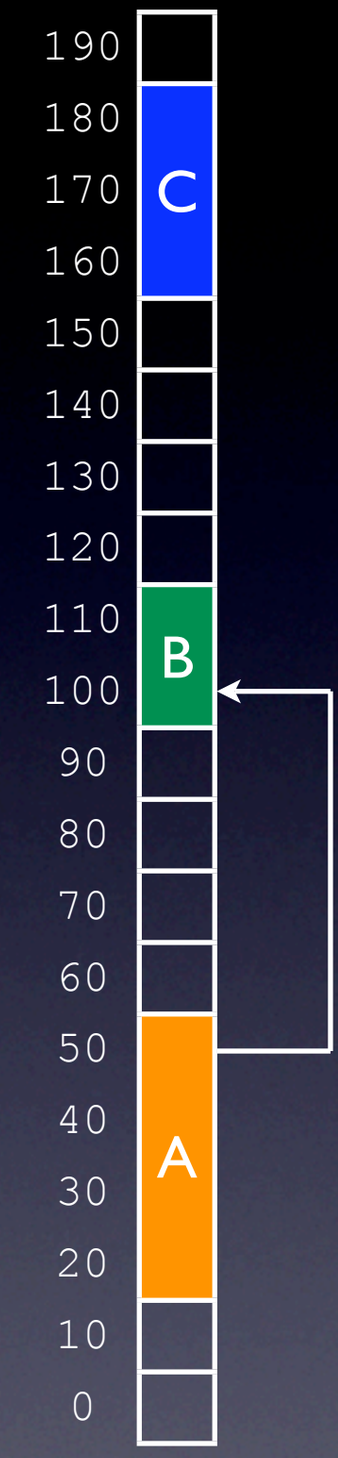
The Problem



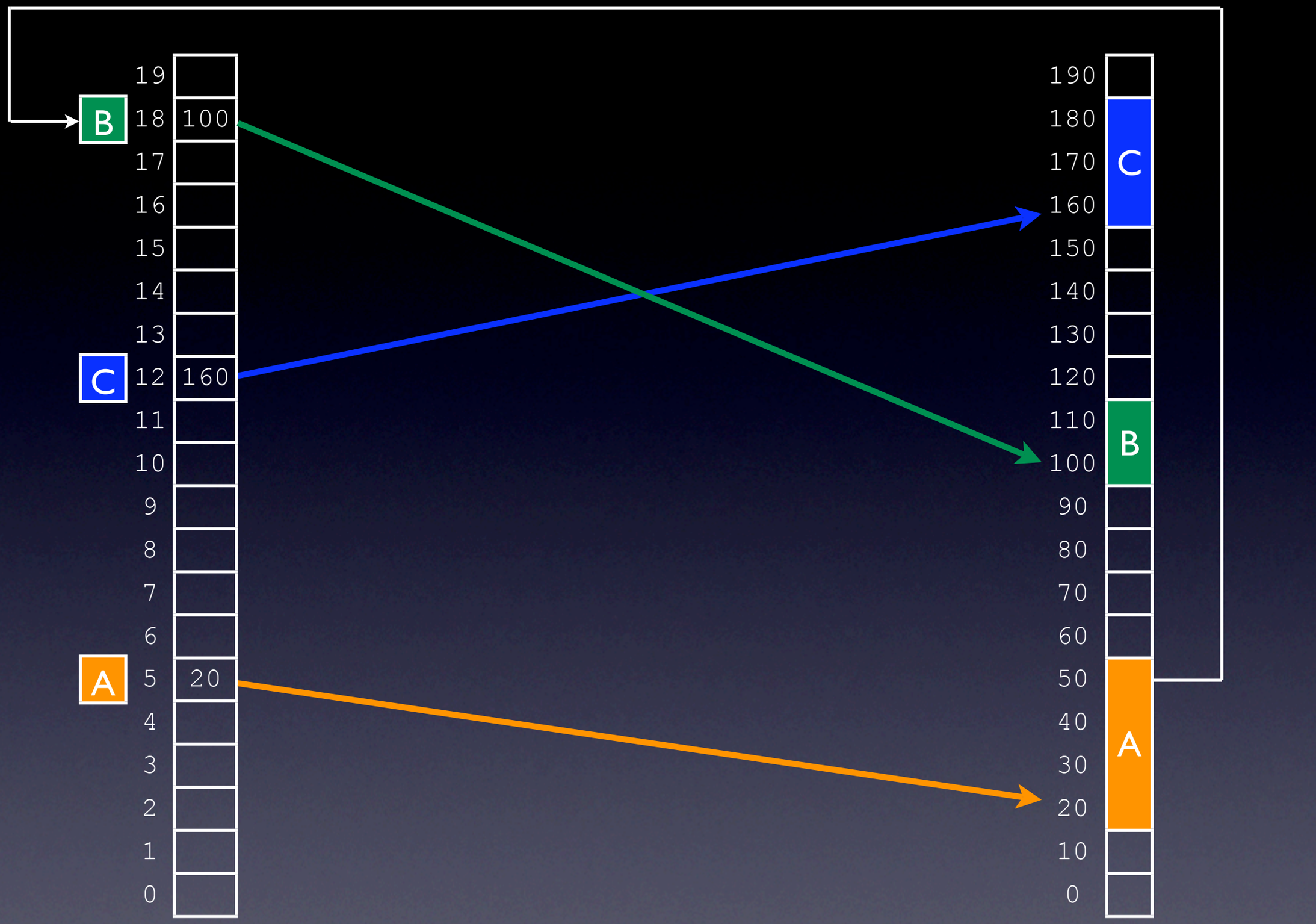
- Fragmentation
  - Compaction
- References
  - Abstract Space

Example:

- There are three objects
- Object A starts at address 20
- Object A needs 40 bytes
- B starts at 100, needs 20 bytes
- C starts at 160, needs 30 bytes
- A contains a reference to B



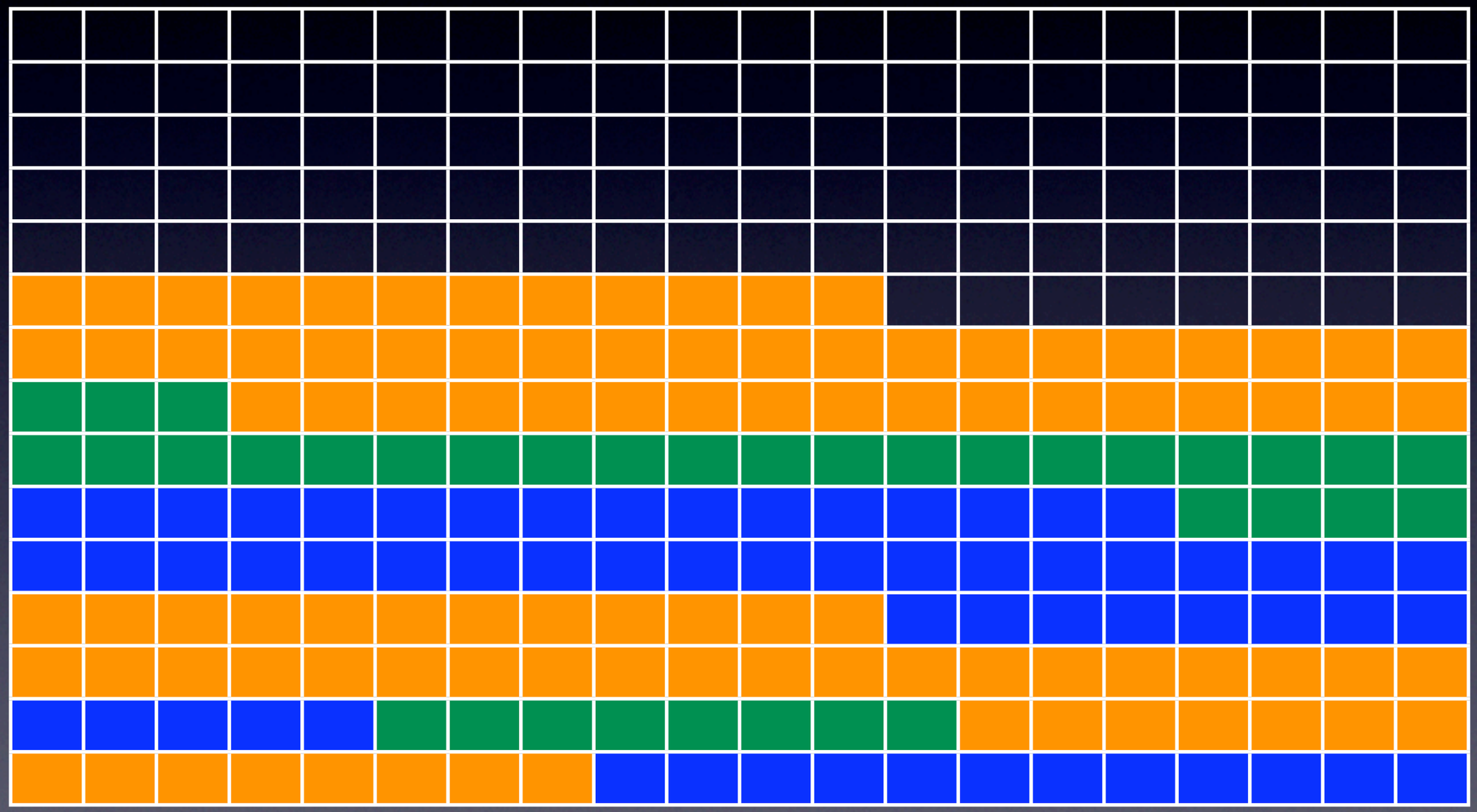
Memory



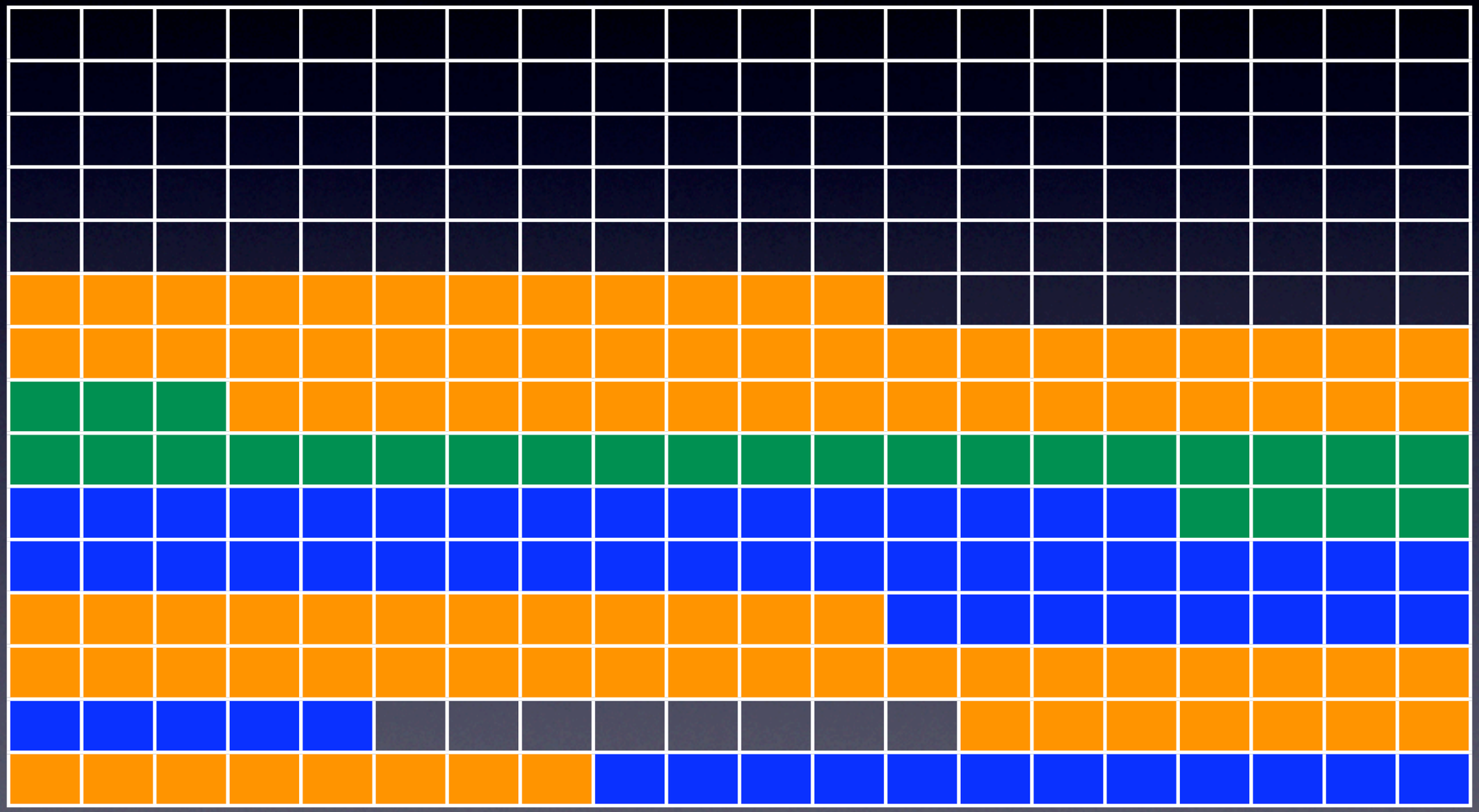
Abstract Space

Concrete Space

Keep It Compact?



Does Not Work!



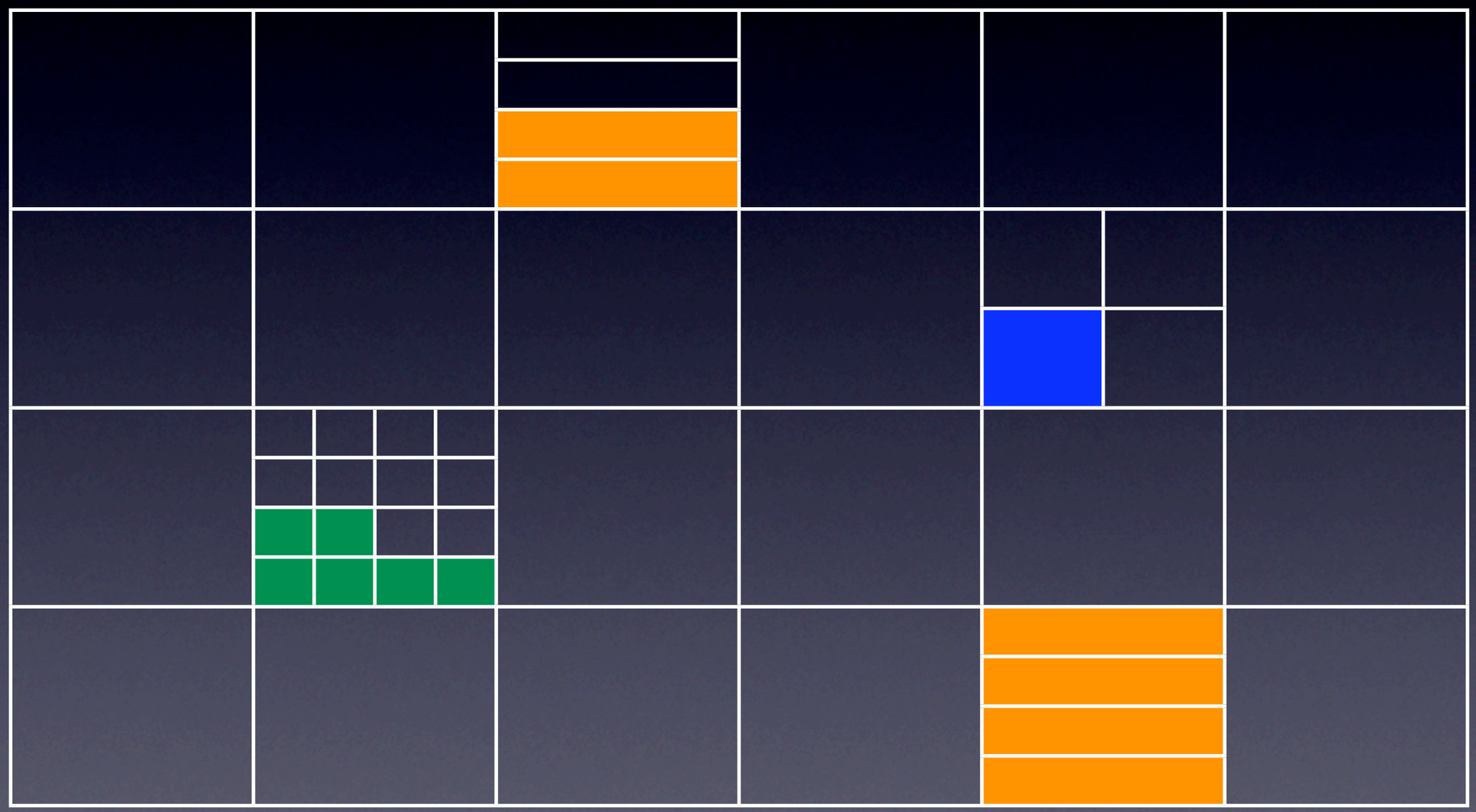
Trade-Off Speed for Memory Fragmentation

Keep Speed and Memory Fragmentation Bounded and Predictable

Partition Memory into Pages

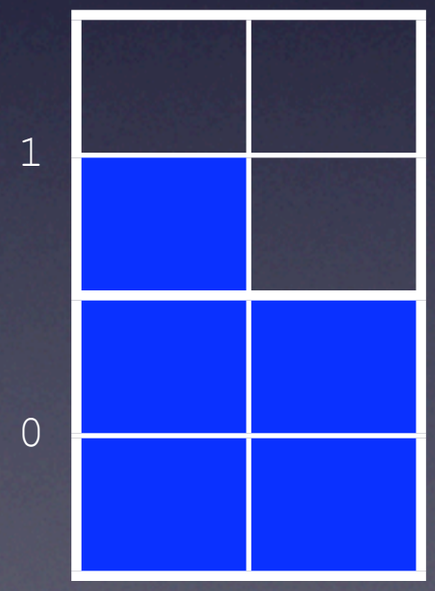
16KB	16KB	16KB	16KB	16KB	16KB
16KB	16KB	16KB	16KB	16KB	16KB
16KB	16KB	16KB	16KB	16KB	16KB
16KB	16KB	16KB	16KB	16KB	16KB

Partition Pages into Blocks

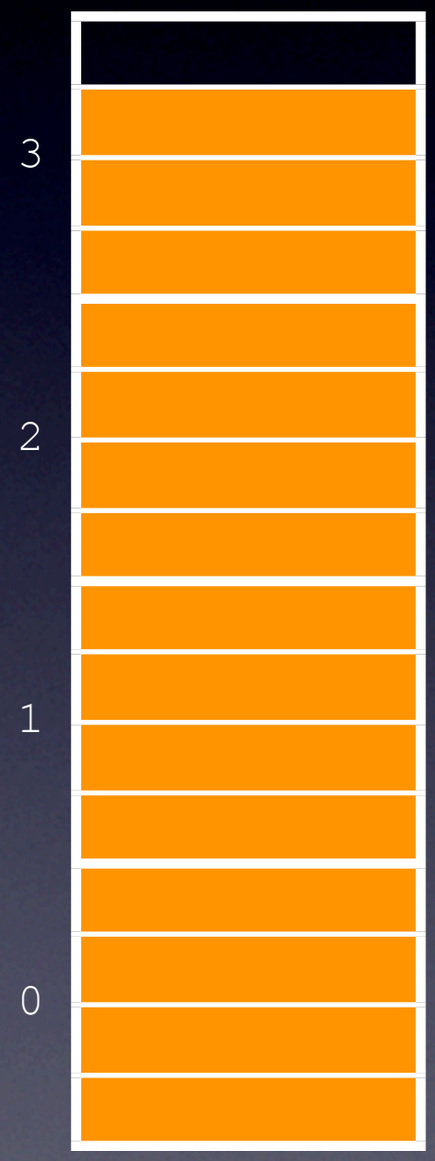


Size-Class Compact

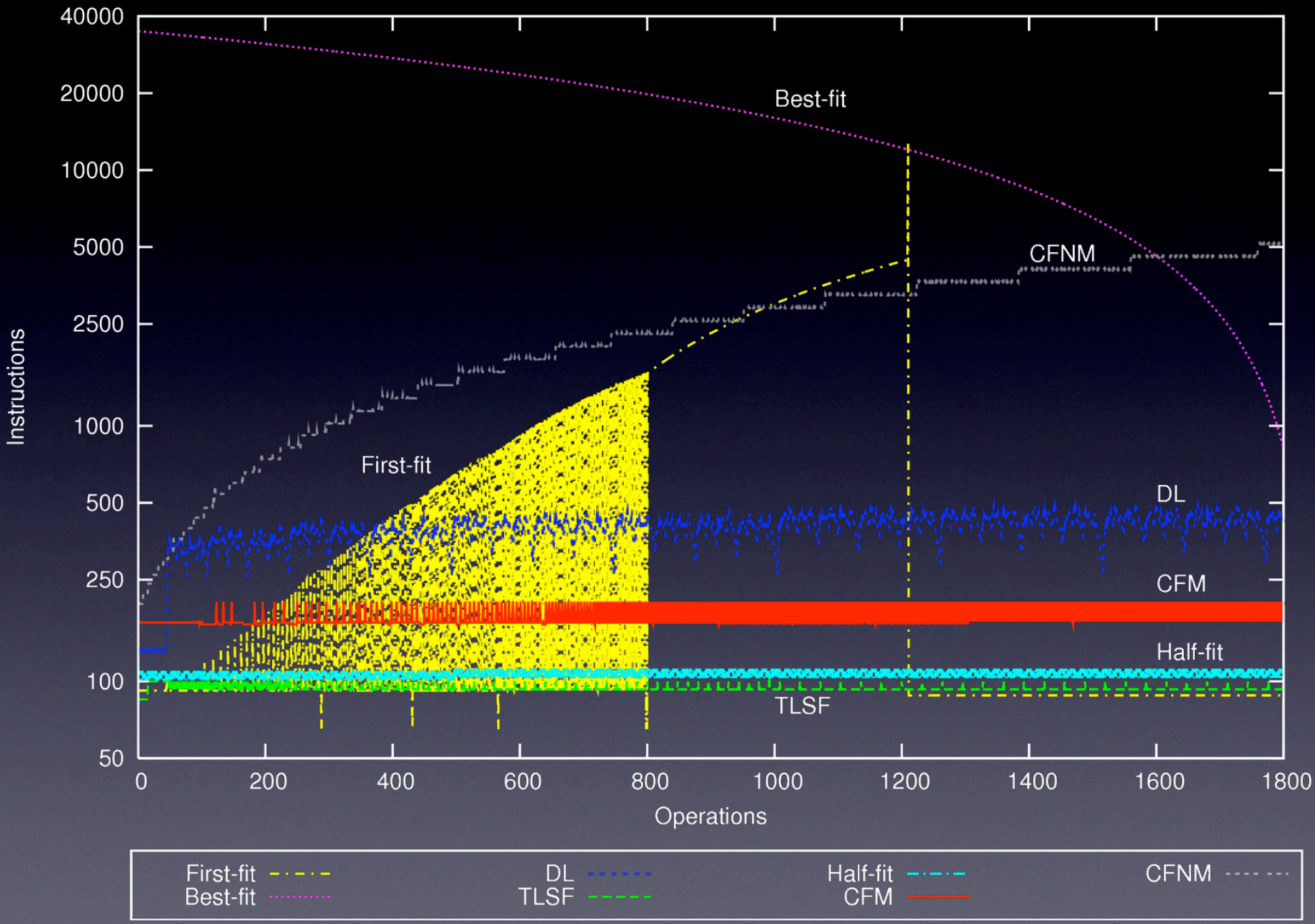
Objects < 32



Objects < 48



Objects < 64

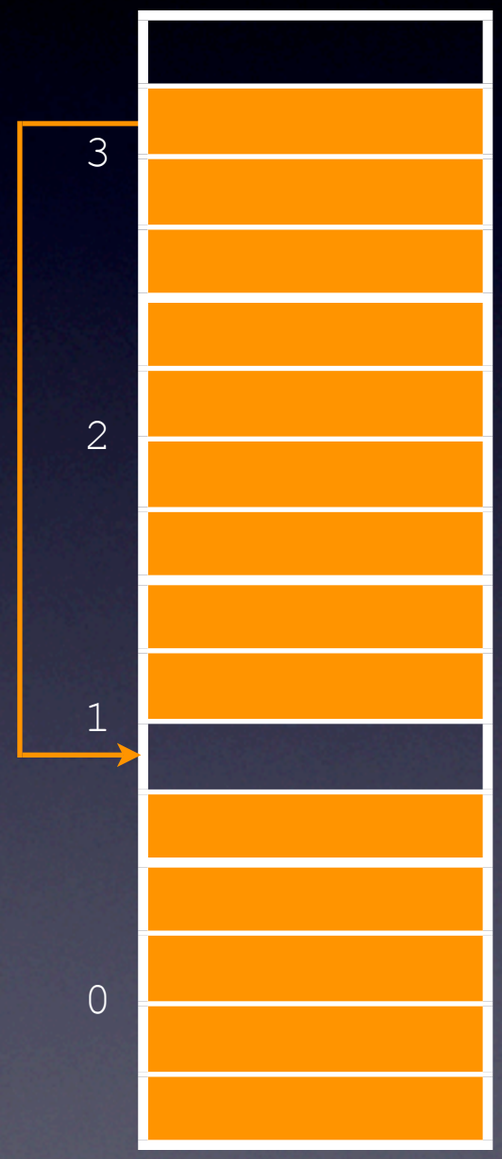

40000
20000
10000
5000
2500
1000
500
250
100
50
Instructions
0
200
400
600
800
1000
1200
1400
1600
1800
Operations
Best-fit
CFNM
First-fit
DL
CFM
Half-fit
TLSF
First-fit
Best-fit
DL
TLSF
Half-fit
CFM
CFNM

“Compact-Fit” (Bounded Compaction)

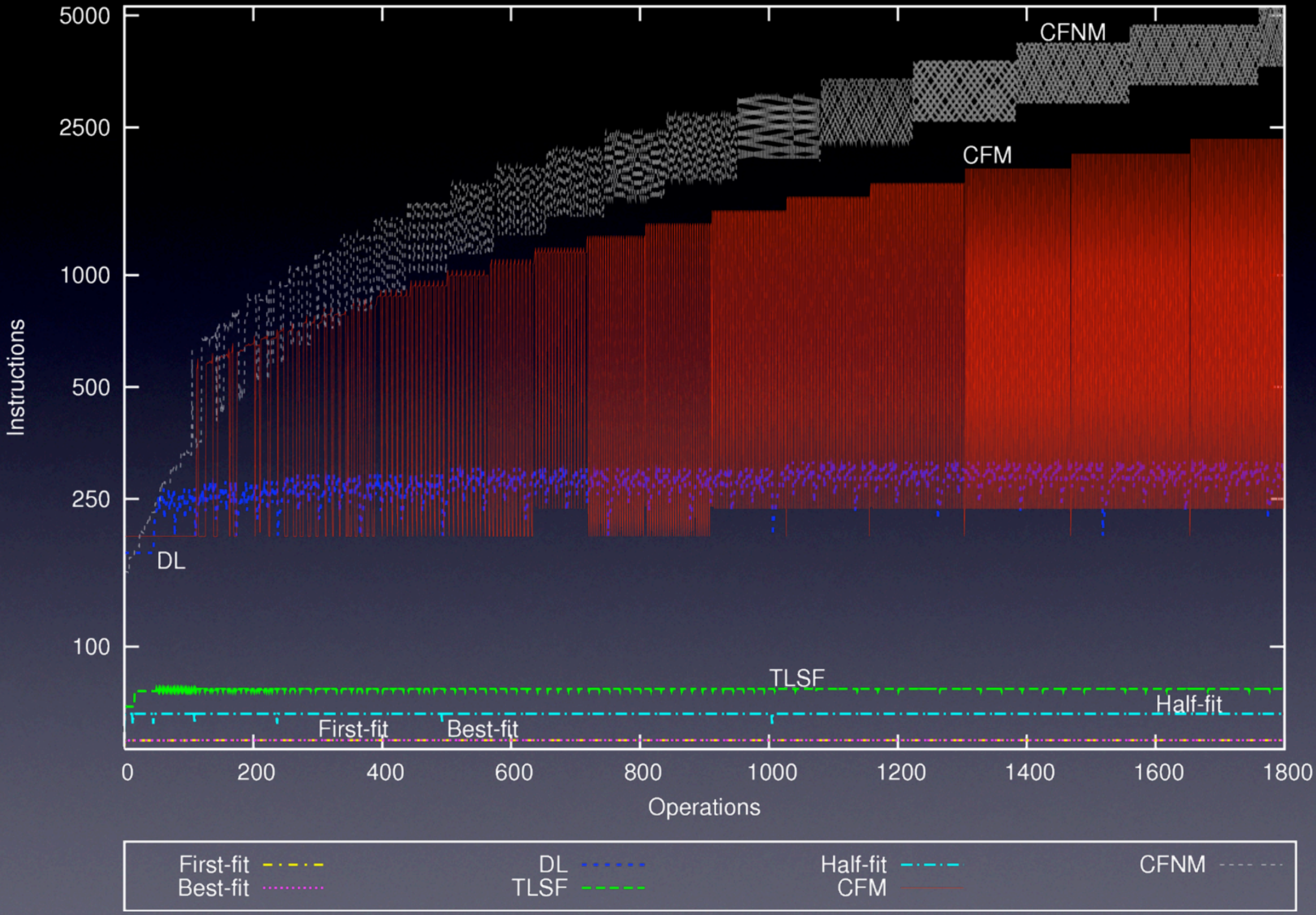
just move ‘last’ object

Objects < 32

Objects < 48



Objects < 64


5000
2500
1000
500
250
100
Instructions
CFNM
CFM
DL
TLSF
Half-fit
First-fit
Best-fit
0
200
400
600
800
1000
1200
1400
1600
1800
Operations
First-fit
Best-fit
DL
TLSF
Half-fit
CFM
CFNM

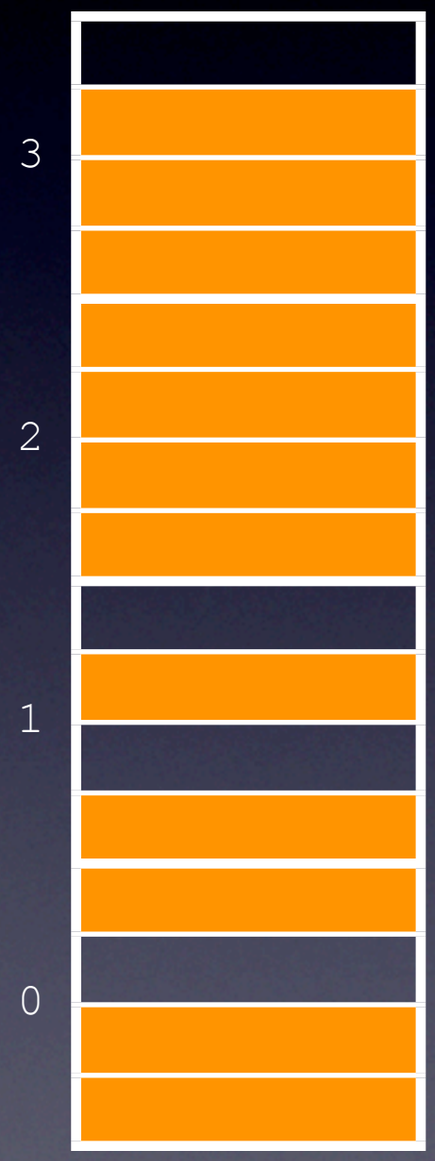
Results I

- malloc(n) takes O(1)
- free(n) takes O(n)
- access takes one indirection
- memory fragmentation bounded in k and predictable in constant time

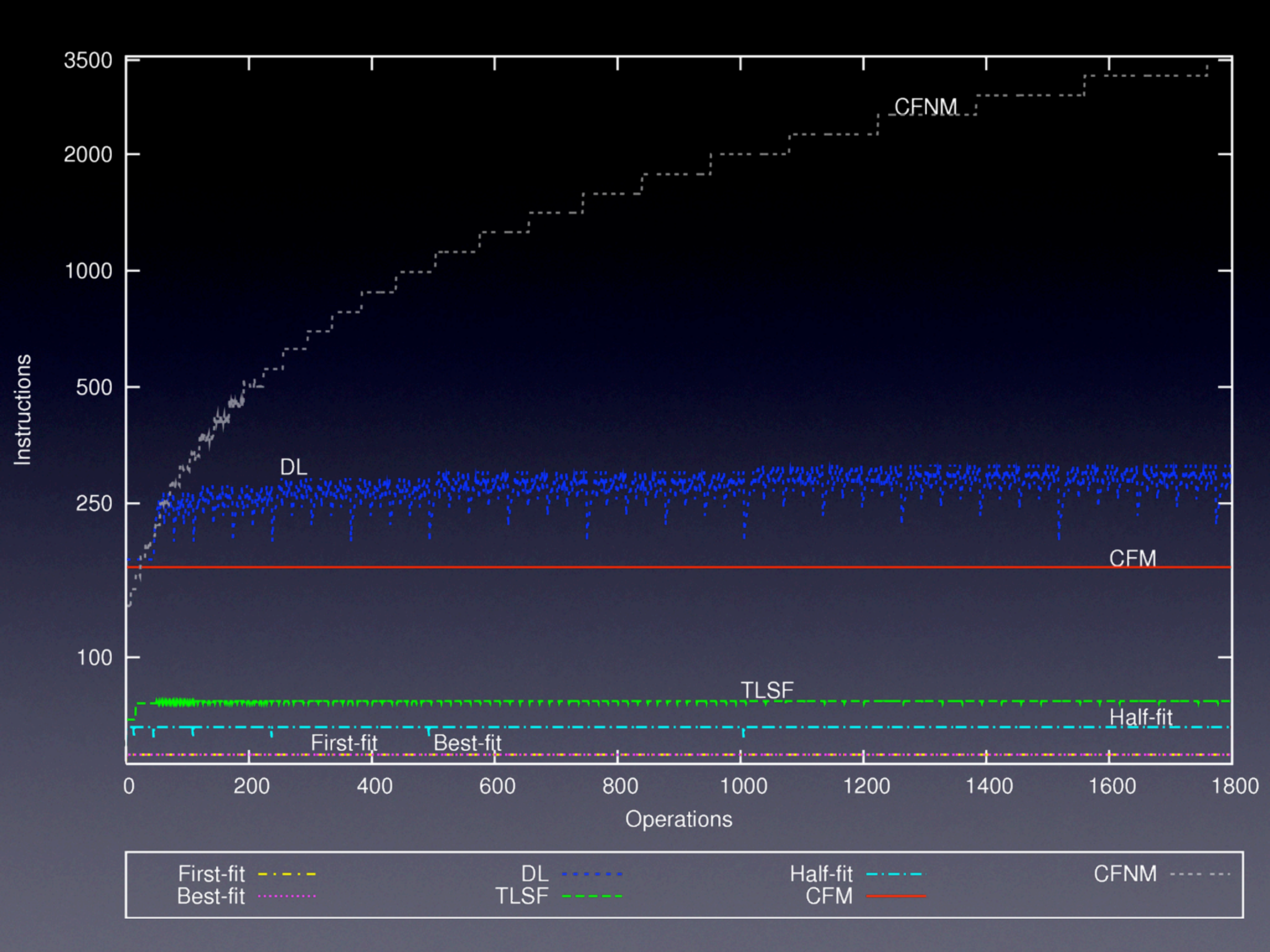
Partial Compaction

Objects < 32

Objects < 48



Objects < 64


3500
2000
1000
500
250
100
Instructions
CFNM
DL
CFM
TLSF
Half-fit
First-fit
Best-fit
0
200
400
600
800
1000
1200
1400
1600
1800
Operations
First-fit
Best-fit
DL
TLSF
Half-fit
CFM
CFNM

Program Analysis

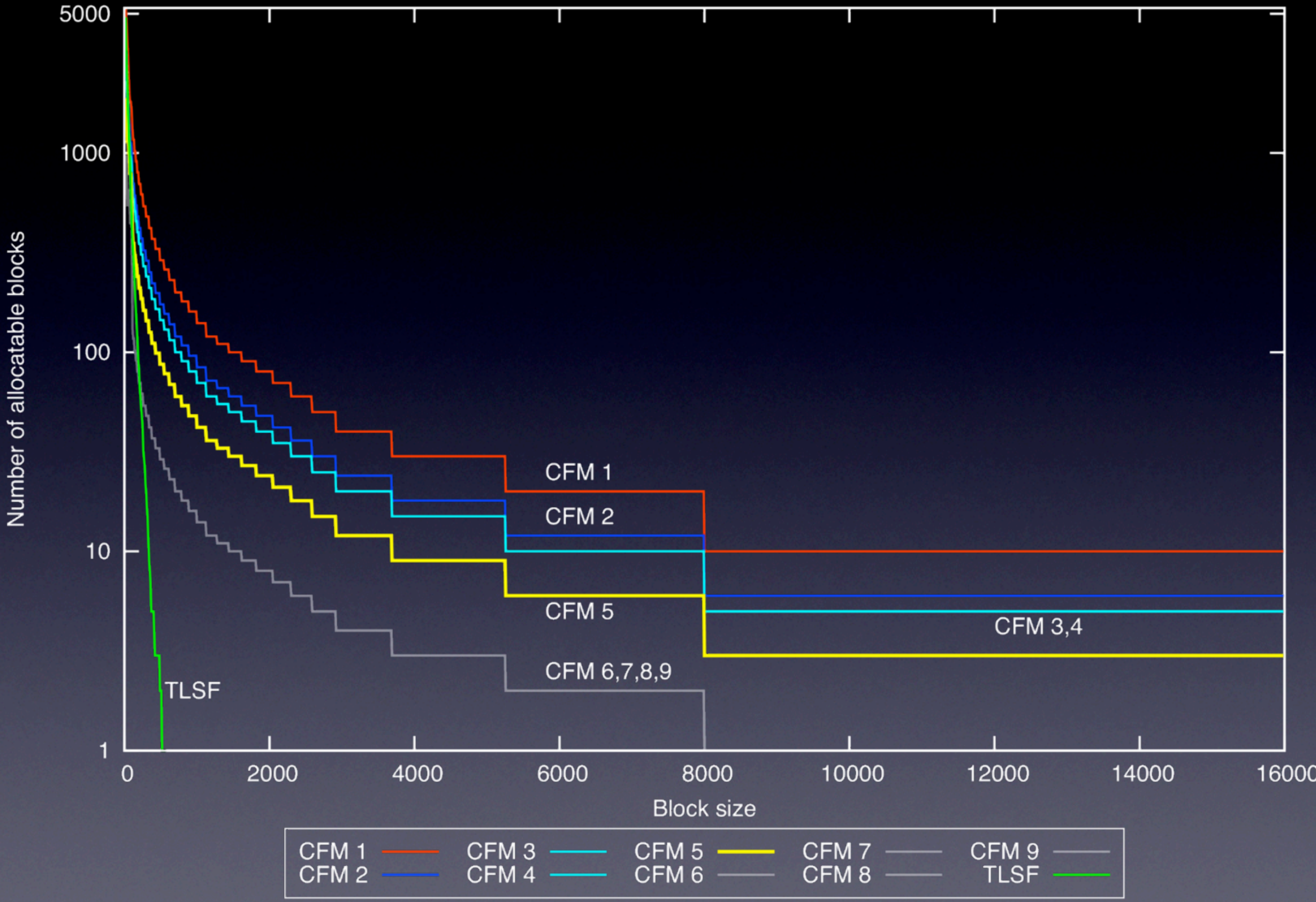
Definition:
Let *k* count deallocations in a given size-class for which no subsequent allocation was done ("k-band mutator").

Proposition:
Each deallocation that happens when *k* < max_number_of_non_full_pages takes constant time.

Results II

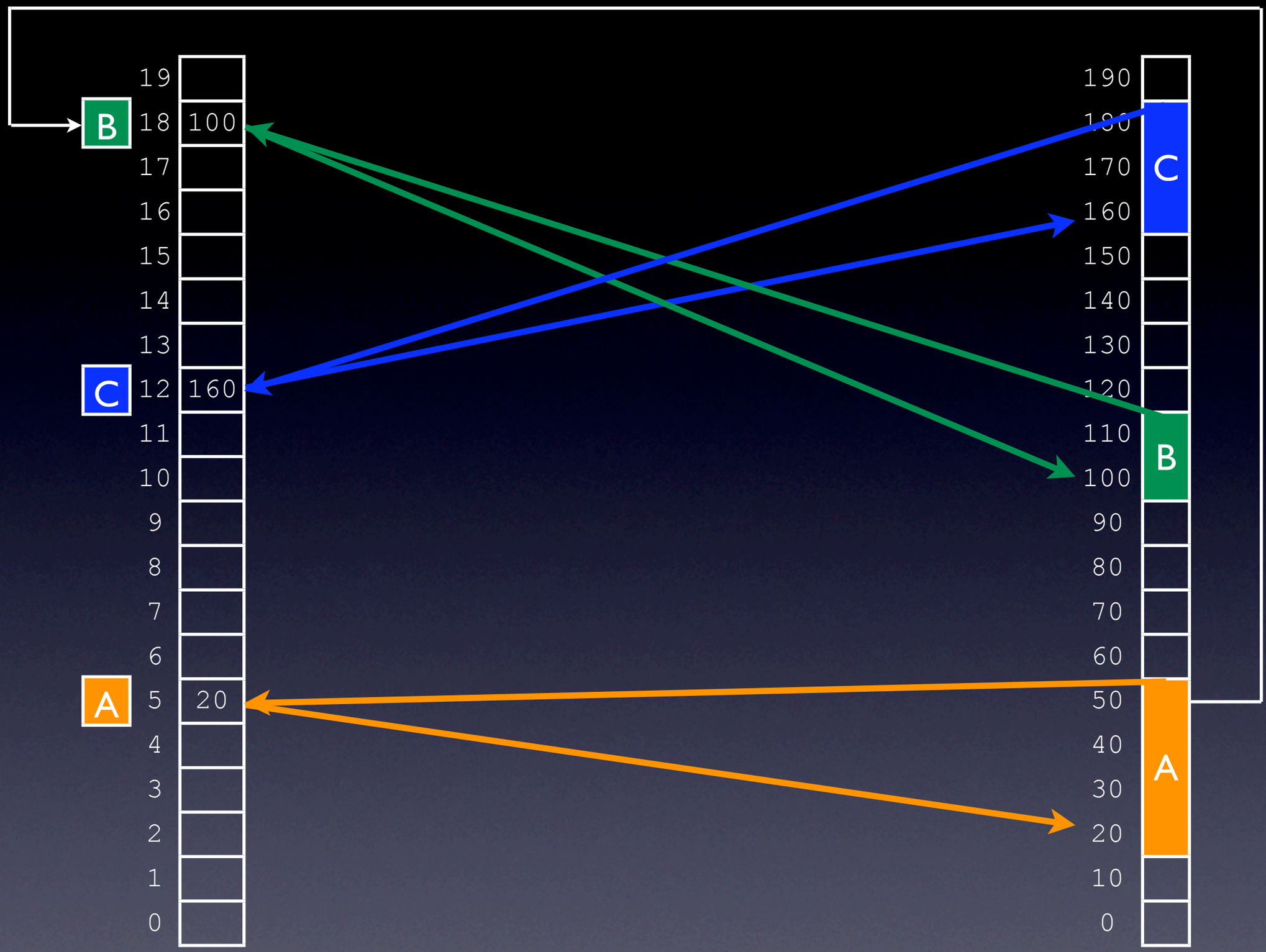
- if mutator stays within k-bands:
  - malloc(n) takes O(1)
  - free(n) takes O(1)
  - access takes one indirection

- memory fragmentation bounded in k and predictable in constant time


5000
1000
100
10
1
Number of allocatable blocks
CFM 1
CFM 2
CFM 5
CFM 3,4
CFM 6,7,8,9
TLSF
0
2000
4000
6000
8000
10000
12000
14000
16000
Block size
CFM 1
CFM 2
CFM 3
CFM 4
CFM 5
CFM 6
CFM 7
CFM 8
CFM 9
TLSF

Two Implementations!

1. Concrete Space = Physical Memory
2. Concrete Space = Virtual Memory

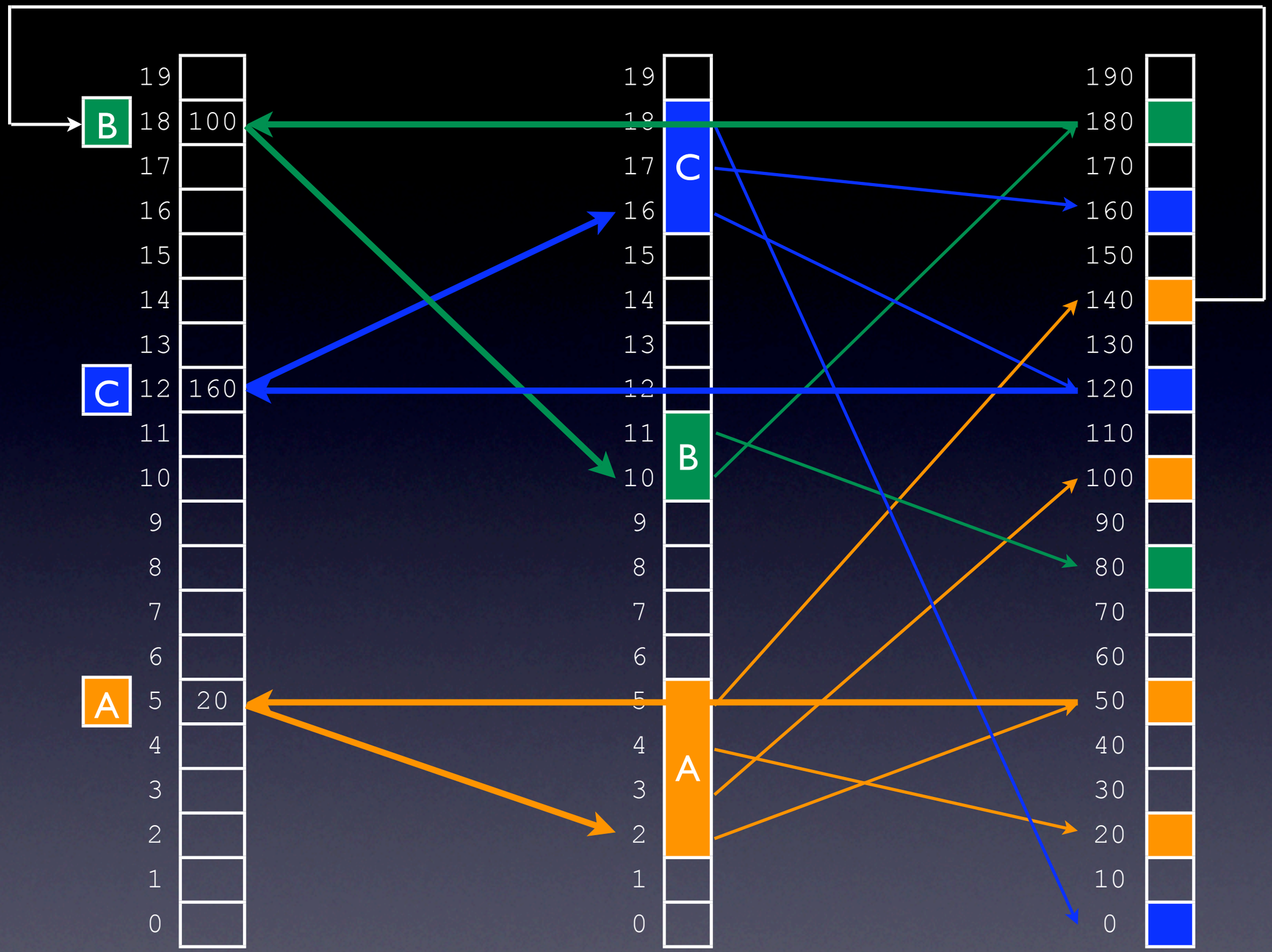


Abstract Space

Physical Memory

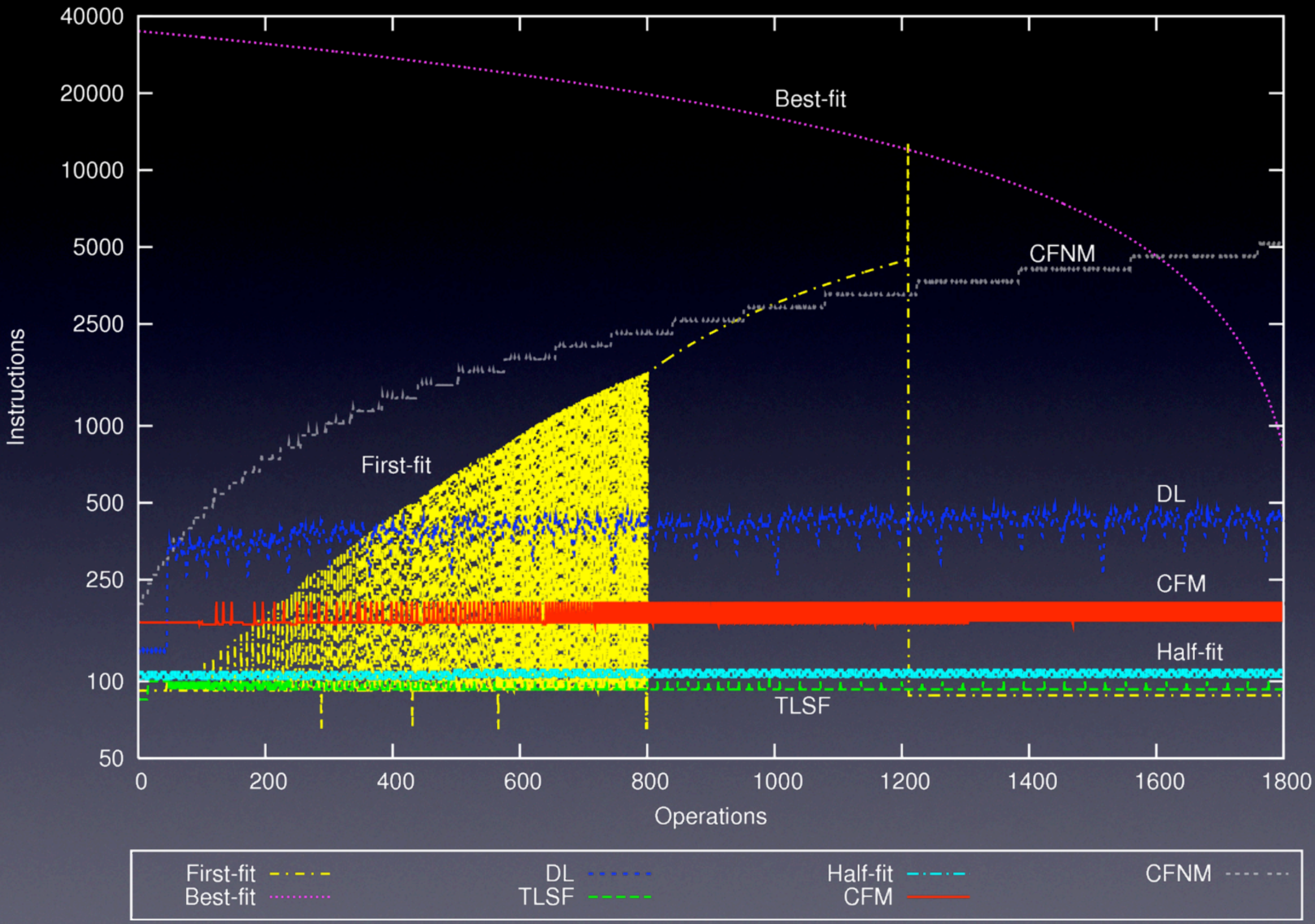
Two Implementations!

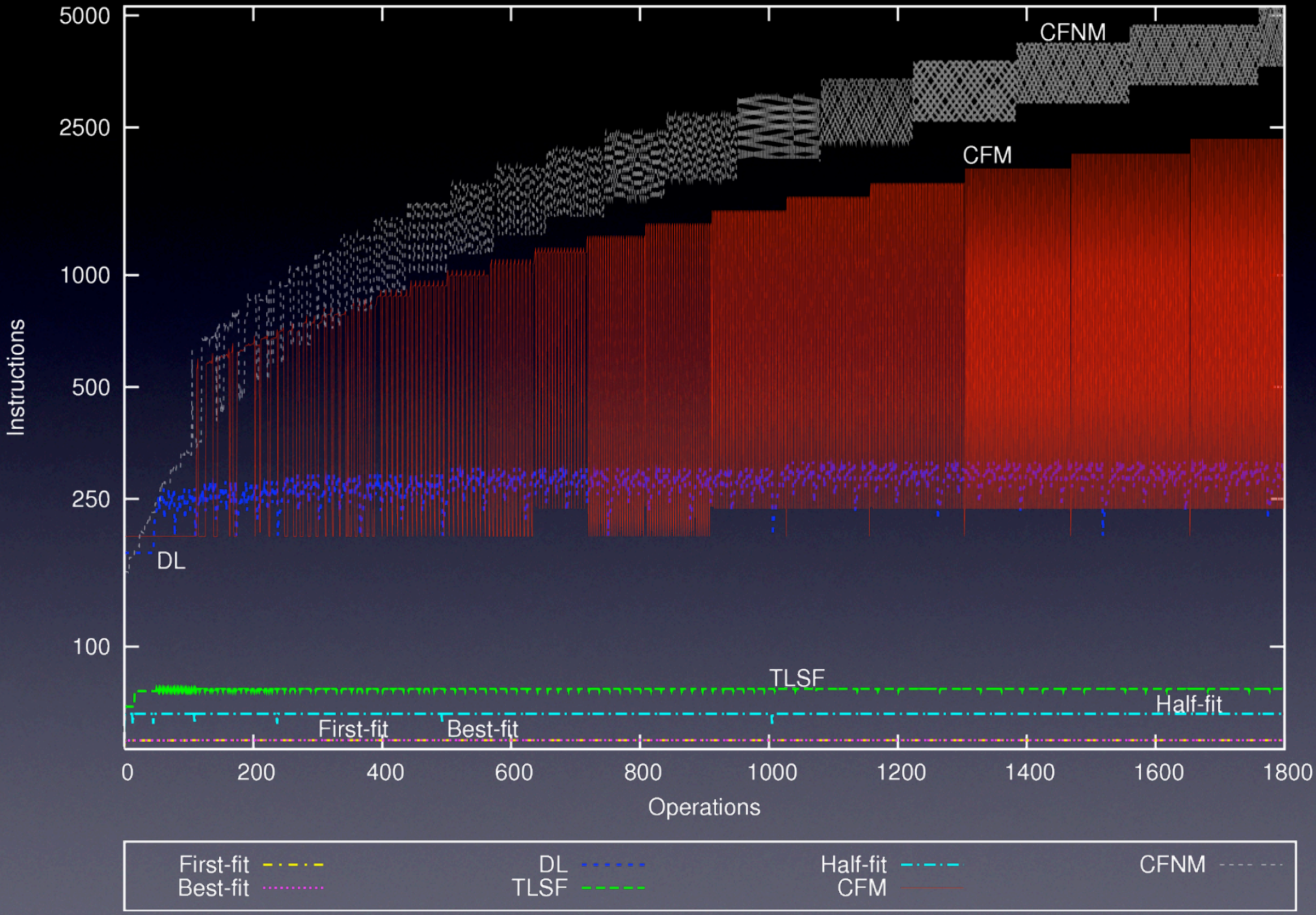
1. Concrete Space = Physical Memory
2. Concrete Space = Virtual Memory



Results III

- malloc(n) takes O(n)
- free(n) takes O(n)
- access takes two indirections
- memory fragmentation bounded in k and predictable in constant time


40000
20000
10000
5000
2500
1000
500
250
100
50
Instructions
0
200
400
600
800
1000
1200
1400
1600
1800
Operations
Best-fit
CFNM
First-fit
DL
CFM
Half-fit
TLSF
First-fit
Best-fit
DL
TLSF
Half-fit
CFM
CFNM

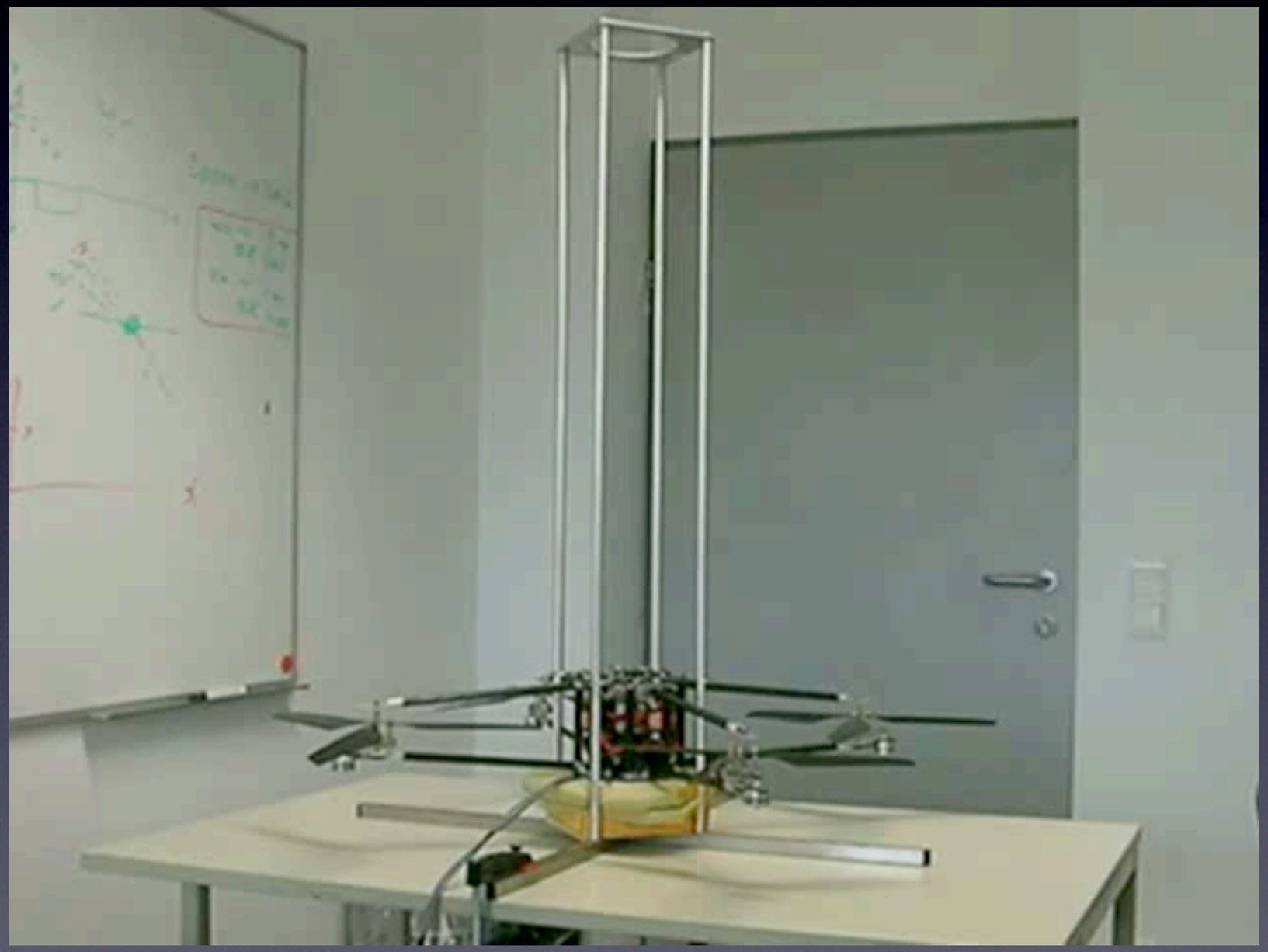

5000
2500
1000
500
250
100
Instructions
CFNM
CFM
DL
TLSF
Half-fit
First-fit
Best-fit
0
200
400
600
800
1000
1200
1400
1600
1800
Operations
First-fit
Best-fit
DL
TLSF
Half-fit
CFM
CFNM

The JAviator

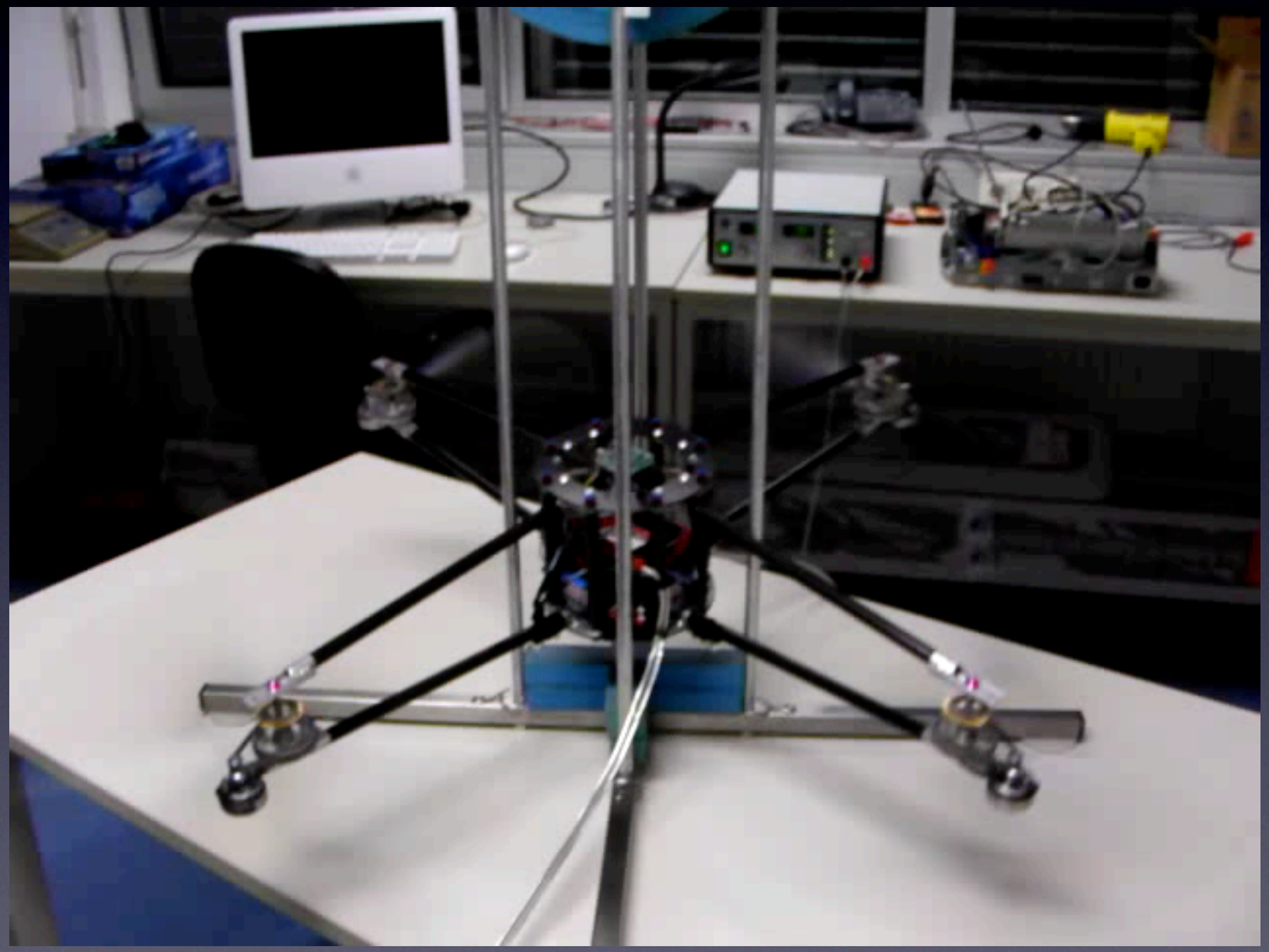
javiator.cs.uni-salzburg.at

Quad-Rotor Helicopter

Oops



Flight Control



Free Flight

Thank you