A Polynomial Spilling Heuristic: Layered Allocation

Boubacar Diouf\textsuperscript{1 2}, Albert Cohen\textsuperscript{1 2}, Fabrice Rastello\textsuperscript{1 2}

\textsuperscript{1}INRIA
\textsuperscript{2}École Normale Supérieure de Paris
\textsuperscript{3}École Normale Supérieure de Lyon
Register Allocation

The register allocation problem maps temporary variables to machine registers

The Allocation/Spilling Problem

- The allocation chooses the register residents
- It also aims at minimizing the load/store overhead

Assignment/Coloring

- The coloring decides which register is used by which variable

Decoupling

- For the moment, let us assume that these two problems can be decoupled
Register Allocation

The register allocation problem maps temporary variables to machine registers

The Allocation/Spilling Problem

- The allocation chooses the register residents
- It also aims at minimizing the load/store overhead

Assignment/Coloring

- The coloring decides which register is used by which variable

Decoupling

- For the moment, let us assume that these two problems can be decoupled
Register Allocation

The register allocation problem maps temporary variables to machine registers.

The Allocation/Spilling Problem

- The allocation chooses the register residents
- It also aims at minimizing the load/store overhead

Assignment/Coloring

- The coloring decides which register is used by which variable

Decoupling

- For the moment, let us assume that these two problems can be decoupled
Register Allocation

The register allocation problem maps temporary variables to machine registers.

The Allocation/Spilling Problem

- The allocation chooses the register residents
- It also aims at minimizing the load/store overhead

Assignment/Coloring

- The coloring decides which register is used by which variable

Decoupling

- For the moment, let us assume that these two problems can be decoupled
A bit of Terminology

- **Maxlive**: the maximum number of simultaneously live variables

- Given $V$ a set of variables of a program and $R$ a number of available registers

Two sub-problems

- The **lowering problem** finds $S$, a subset of $V$, of minimum cost to spill in order to decrease maxlive by a small number

- The **single layer allocation problem** finds $A$, a subset of $V$, of maximum cost to allocate to a small number of registers

Two Approaches to the Allocation Problem

- The **layered allocation** incrementally solves the single layer allocation problem until the sum of the used registers reaches $R$

- The **incremental lowering** incrementally solves the lowering problem until maxlive reaches $R$
A bit of Terminology

- **Maxlive**: the maximum number of simultaneously live variables
- Given $V$ a set of variables of a program and $R$ a number of available registers

Two sub-problems

- The **lowering problem** finds $S$, a subset of $V$, of minimum cost to spill in order to decrease maxlive by a small number
- The **single layer allocation problem** finds $A$, a subset of $V$, of maximum cost to allocate to a small number of registers

Two Approaches to the Allocation Problem

- The **layered allocation** incrementally solves the single layer allocation problem until the sum of the used registers reaches $R$
- The **incremental lowering** incrementally solves the lowering problem until maxlive reaches $R$
A bit of Terminology

- **Maxlive**: the maximum number of simultaneously live variables
- Given $V$ a set of variables of a program and $R$ a number of available registers

Two sub-problems

- The **lowering problem** finds $S$, a subset of $V$, of minimum cost to spill in order to decrease maxlive by a small number
- The **single layer allocation problem** finds $A$, a subset of $V$, of maximum cost to allocate to a small number of registers

Two Approaches to the Allocation Problem

- The **layered allocation** incrementally solves the single layer allocation problem until the sum of the used registers reaches $R$
- The **incremental lowering** incrementally solves the lowering problem until maxlive reaches $R$
A bit of Terminology

- **Maxlive**: the maximum number of simultaneously live variables
- Given $V$ a set of variables of a program and $R$ a number of available registers

Two sub-problems

- The **lowering problem** finds $S$, a subset of $V$, of minimum cost to spill in order to decrease maxlive by a small number
- The **single layer allocation problem** finds $A$, a subset of $V$, of maximum cost to allocate to a small number of registers

Two Approaches to the Allocation Problem

- The **layered allocation** incrementally solves the single layer allocation problem until the sum of the used registers reaches $R$
- The **incremental lowering** incrementally solves the lowering problem until maxlive reaches $R$
Lowering vs. Layering

![Graph showing number of live variables vs. program points comparing Lowering and Layering approaches.](image-url)
Lowering vs. Layering

Program points
Number of live variables
maxlive
R
0

Program points
Lowering vs. Layering

Number of live variables

maxlive

maxlive - small

R

0

Program points
Lowering vs. Layering

Spilling some variables in order to lower maxlive by small is NP-complete
Even if small = 1!

Number of live variables

maxlive

maxlive - small

R

0

Program points
Lowering vs. Layering

Number of live variables

maxlive

R

small

0

Program points

Layered Allocation
Allocating variables to a small number of registers is polynomial!
Lowering vs. Layering

Proceeding by layered allocation is still polynomial!
Lowering vs. Layering

Proceeding by layered allocation is still polynomial!
Lowering vs. Layering

Number of live variables

maxlive
maxlive - small
R
small
0

Program points

NP-complete
Polynomial

Layered Allocation
Why should the Layered Allocation be Close to Optimal?

- Let us assume that we have a program $P$

- When $R + 1$ registers are available, let us call $SPILL^P_{R+1}$ the optimal set of variables to spill to make a coloring possible

- For most programs, $SPILL^P_{R+1} \subset SPILL^P_R$ \[Diouf’10\]

- Hence, for most programs, $ALLOC^P_R \subset ALLOC^P_{R+1}$
Taxonomy of the Approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Complexity</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocation/Spilling</td>
<td>NP-complete</td>
<td>Optimal</td>
</tr>
<tr>
<td>Layered Allocation</td>
<td>Polynomial</td>
<td>Close to optimal</td>
</tr>
<tr>
<td>Incremental lowering-optimal</td>
<td>NP-complete</td>
<td>???</td>
</tr>
<tr>
<td>Incremental lowering-heuristic</td>
<td>Polynomial</td>
<td>Not-optimal</td>
</tr>
</tbody>
</table>

To make it clear!

- The Allocation problem is NP-complete
- The Layered allocation is a heuristic that is close to optimal allocation
- We are not turning an NP-complete problem into a polynomial one
To make it clear!

- The Allocation problem is NP-complete
- The Layered allocation is a heuristic that is close to optimal allocation
- We are not turning an NP-complete problem into a polynomial one
## Taxonomy of the Approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Complexity</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocation/Spilling</td>
<td>NP-complete</td>
<td>Optimal</td>
</tr>
<tr>
<td>Layered Allocation</td>
<td>Polynomial</td>
<td>Close to optimal</td>
</tr>
<tr>
<td>Incremental lowering-optimal</td>
<td>NP-complete</td>
<td>???</td>
</tr>
<tr>
<td>Incremental lowering-heuristic</td>
<td>Polynomial</td>
<td>Not-optimal</td>
</tr>
</tbody>
</table>

To make it clear!

- The Allocation problem is NP-complete
- The Layered allocation is a heuristic that is close to optimal allocation
- We are not turning an NP-complete problem into a polynomial one
Taxonomy of the Approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Complexity</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocation/Spilling</td>
<td>NP-complete</td>
<td>Optimal</td>
</tr>
<tr>
<td>Layered Allocation</td>
<td>Polynomial</td>
<td>Close to optimal</td>
</tr>
<tr>
<td>Incremental lowering-optimal</td>
<td>NP-complete</td>
<td>???</td>
</tr>
<tr>
<td>Incremental lowering-heuristic</td>
<td>Polynomial</td>
<td>Not-optimal</td>
</tr>
</tbody>
</table>

To make it clear!

- The Allocation problem is NP-complete
- The Layered allocation is a heuristic that is close to optimal allocation
- We are not turning an NP-complete problem into a polynomial one
Outline

Introduction

Layered Approach
  Layered-Heuristic Allocation: General Graphs
  Layered-Optimal Allocation: Chordal Graphs

Experimental Evaluation

Conclusion
Two Heuristics for the Spilling Problem

Input:
1. A register allocation problem where each variable has an estimated spill cost
2. A number of available registers

Objective:
We want to perform an allocation that minimizes the cost of all the spilled variables

Two graph-based solutions:
- The general approach: Layered-Heuristic Register Allocator
- The SSA-based approach: Layered-Optimal Register Allocator
The General Approach

Given an interference graph of a program and $R$ available registers (colors)

1. Assume that we have one register

2. We approximate the set of nodes of maximum cost/weight to allocate with one register: a layer. This layer is an independent set.

3. Remove the nodes of the layer from the graph at the next iteration

Repeat these instructions until we reach $R$ or we allocate all the variables
How the Layered-Heuristic Works

2 available registers

Layered Allocation
How the Layered-Heuristic Works

Variables sorted by decreasing cost: \(a, e, c, b, d, f\)

Layered Allocation

2 available registers
How the Layered-Heuristic Works

Variables sorted by decreasing cost: \(a, e, c, b, d, f\)

I-Set-1: \{a\}

2 available registers
How the Layered-Heuristic Works

Variables sorted by decreasing cost: $e, c, b, d, f$

I-Set-1: \{a, e\}

2 available registers
How the Layered-Heuristic Works

Variables sorted by decreasing cost: c, b, d, f

I-Set-1: \{a,e\}

2 available registers

Layered Allocation
How the Layered-Heuristic Works

Variables sorted by decreasing cost: b, d, f

I-Set-1: {a, e}

I-Set-2: {c, f}

I-Set-3: {b, d}
How the Layered-Heuristic Works

Variables sorted by decreasing cost: b, d

I-Set-1: \{a,e\}
I-Set-2: \{c,f\}
How the Layered-Heuristic Works

Variables sorted by decreasing cost:

I-Set-1: \{a,e\}
I-Set-2: \{c,f\}
I-Set-3: \{b,d\}
How the Layered-Heuristic Works

Variables sorted by decreasing cost:

I-Set-1: \{a,e\}

I-Set-2: \{c,f\}

I-Set-3: \{b,d\}

I-Sets sorted by decreasing cost: I-Set-1, I-Set-3, I-Set-2

2 available registers
How the Layered-Heuristic Works

Variables sorted by decreasing cost:

I-Set-1: \{a,e\}
I-Set-2: \{c,f\}
I-Set-3: \{b,d\}

I-Sets sorted by decreasing cost: I-Set-1, I-Set-3, I-Set-2

2 available registers

The cost of the allocation is 5
How the Layered-Heuristic Works

Variables sorted by decreasing cost:

- I-Set-1: \{a, e\}
- I-Set-2: \{c, f\}
- I-Set-3: \{b, d\}

I-Sets sorted by decreasing cost: I-Set-1, I-Set-3, I-Set-2

The cost of the allocation is 5
Outline

Introduction

Layered Approach
  - Layered-Heuristic Allocation: General Graphs
  - Layered-Optimal Allocation: Chordal Graphs

Experimental Evaluation

Conclusion
SSA-based Interference Graphs

The interference graph of an SSA-based program is chordal

1. The allocation problem can be decoupled from the coloring problem thanks to maxlive

2. Hence, the maximum weighted independent set can be found optimally [Frank’75]
The Maximum Weighted Independent Set Algorithm

Weighted graph

Layered Allocation
The Maximum Weighted Independent Set Algorithm

iteration | a | f | d | e | b | g | c | red vertices
--- | --- | --- | --- | --- | --- | --- | --- | ---
- | 1 | 6 | 5 | 2 | 2 | 1 | 2 | Ø

Red vertices
The Maximum Weighted Independent Set Algorithm

Weighted graph

<table>
<thead>
<tr>
<th>iteration</th>
<th>a</th>
<th>f</th>
<th>d</th>
<th>e</th>
<th>b</th>
<th>g</th>
<th>c</th>
<th>red vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>Ø</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
<td>a</td>
</tr>
</tbody>
</table>

Red vertices
The Maximum Weighted Independent Set Algorithm

<table>
<thead>
<tr>
<th>iteration</th>
<th>a</th>
<th>f</th>
<th>d</th>
<th>e</th>
<th>b</th>
<th>g</th>
<th>c</th>
<th>red vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>Ø</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>-3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>f,</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>b,</td>
<td>f,a</td>
<td></td>
</tr>
</tbody>
</table>

Red vertices
The Maximum Weighted Independent Set Algorithm

Weighted graph

<table>
<thead>
<tr>
<th>iteration</th>
<th>a</th>
<th>f</th>
<th>d</th>
<th>e</th>
<th>b</th>
<th>g</th>
<th>c</th>
<th>red vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>Ø</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>-3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td>f, a</td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>b,f,a</td>
</tr>
</tbody>
</table>

Red vertices

<table>
<thead>
<tr>
<th>iteration</th>
<th>red vertices</th>
<th>blue vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>b,f,a</td>
<td>Ø</td>
</tr>
</tbody>
</table>

Blue vertices
The Maximum Weighted Independent Set Algorithm

**Weighted graph**

<table>
<thead>
<tr>
<th>iteration</th>
<th>a</th>
<th>f</th>
<th>d</th>
<th>e</th>
<th>b</th>
<th>g</th>
<th>c</th>
<th>red vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>Ø</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>-3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td>f, a</td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>b,f,a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>iteration</th>
<th>iteration</th>
<th>red vertices</th>
<th>blue vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>b,f,a</td>
<td>Ø</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>f,a</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>f,a</td>
<td>b</td>
<td></td>
</tr>
</tbody>
</table>

**Red vertices**

**Blue vertices**

Layered Allocation
The Maximum Weighted Independent Set Algorithm

Layered Allocation
How the Layered-Optimal Allocator Works

2 available registers

Layered Allocation
How the Layered-Optimal Allocator Works

2 available registers
How the Layered-Optimal Allocator Works

2 available registers

[Diagram showing node allocation]
How the Layered-Optimal Allocator Works

Layered Allocation

2 available registers

- Blue square (Available)
- Red square (Occupied)
A First Improvement: Weights Bias

Allocated variables: \{f, b, d, g\}

Allocation-Cost = 14

2 available registers
A First Improvement: Weights Bias

Allocated variables: {f, b, d, g}
Allocation-Cost = 14

2 available registers
A First Improvement: Weights Bias

Allocated variables: \{f, b, d, g\}
Allocation-Cost = 14

Allocated variables: \{f, c\}
A First Improvement: Weights Bias

Allocated variables: \{f, b, d, g\}
Allocation-Cost = 14

Allocated variables: \{f, c, d, b\}
A First Improvement: Weights Bias

Allocated variables: \{f, b, d, g\}
Allocation-Cost = 14

Allocated variables: \{f, c, d, b\}
Allocation-Cost = 15

2 available registers
A Second Improvement: A Fixed Point Iteration

Layered Allocation

2 available registers

Layered Allocation
A Second Improvement: A Fixed Point Iteration

2 available registers

Layered Allocation
A Second Improvement: A Fixed Point Iteration

Layered Allocation

2 available registers
A Second Improvement: A Fixed Point Iteration

Layered Allocation

2 available registers

A maximal clique
A Second Improvement: A Fixed Point Iteration

2 available registers

Layered Allocation
A Second Improvement: A Fixed Point Iteration

2 available registers

Layered Allocation
A Second Improvement: A Fixed Point Iteration

2 available registers

Layered Allocation
Outline

Introduction

Layered Approach

Layered-Heuristic Allocation: General Graphs
Layered-Optimal Allocation: Chordal Graphs

Experimental Evaluation

Conclusion
Evaluating the Layered-Heuristic Allocator

Architectures

- x86

Benchmarks extracted from JikesRVM

- SPEC JVM 98

Algorithms

- LS: the linear scan implemented in JikesRVM
- BLS: a variant of the Belady’s furthest-first
- GC: the Chaitin-Briggs optimistic graph coloring
- Optimal: an ILP-based Allocator
- LH: Layered Heuristic
Evaluating the Layered-Heuristic Allocator
Evaluating the Layered-Heuristic Allocator

Layered Approach

Evaluation

Layered Allocation
Evaluating the Layered-Optimal Allocator

Architectures

- ARMv7
- ST231

Benchmarks

- eembc
- lao-kernels
- SPEC CPU 2000int

Algorithms

- GC: the Chaitin-Briggs optimistic graph coloring
- Optimal: an ILP-based Allocator
- L: our baseline Layered-Optimal approach
- BL: the biased variant of our Layered-Optimal
- FPL: the fixed-point variant of our Layered-Optimal
- BFPL: the biased and fixed-point variant of our Layered-Optimal
Evaluating the Layered-Optimal Allocator

Layered Allocation
Evaluating the Layered-Optimal Allocator

Layered Allocation
Outline

Introduction

Layered Approach
  Layered-Heuristic Allocation: General Graphs
  Layered-Optimal Allocation: Chordal Graphs

Experimental Evaluation

Conclusion
Conclusion

Contributions

- Layered allocation: polynomial and close to optimal allocation
- Iteratively allocate instead of (classical) iteratively spilling
- The approach works on general graphs and on SSA-based graphs