On the Decidability of Phase Ordering Problem in Optimizing Compilation

S.Touati D.Barthou

U. of Versailles

Computing Frontiers'06

Outline

Computing Frontier: Compiler Construction

- Position of the problem and context
- Phase ordering problem
- Finding optimal values for optimization parameters
- Concluding remarks

Position of the Problem: Generating optimal code

Many individual optimizations exist.

Effect of individual optimizations:

- Depends on parameters
- Not always beneficial on performance
- Complex interactions with others

Position of the Problem: Generating optimal code

Many individual optimizations exist.

Effect of individual optimizations:

- Depends on parameters
- Not always beneficial on performance
- Complex interactions with others

Searching the best optimizations and parameters to generate optimal programs

Automatic, compilation time:

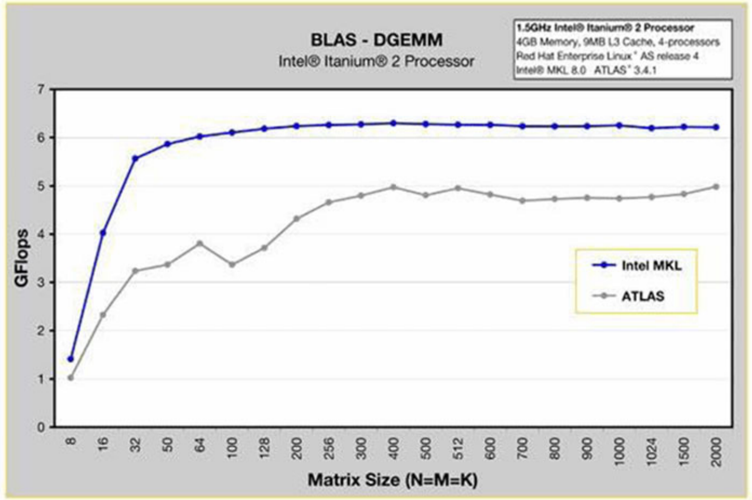
- Heuristics drive optimization phase ordering in compilers
- Exhaustive search [Cooper], limiting the number of optimizations
- Iterative compilation: search for a 'good' solution.

Performed manually or by hardware:

- Source to source transformations
- `pragmas` or a meta-language
- Trust hardware mecanisms to perform important optimizations ! (eg. OoO)

Position of the Problem: Generating optimal code

Difficult problem: automatic optimization far away from manual



A formalization is needed to better understand the bottleneck

Context: Which Optimality ?

Optimization Issue

Is there an algorithm that builds from a program $\mathcal{P}$ an optimal program $\mathcal{P}^*$ semantically equivalent ?

- Optimality does not exists for any program input I [Schwiegelshohn *et al.*]
- Semantically equivalent: $\mathcal{P}^*$ must execute correctly on all inputs
- Optimality according to a performance model (or execution time).
- Objective: obtain performance lower than some given bound

Context: Which Performance Model ?

Any performance model

- Statistical Linear Regression Models
- Static Algorithmic Models
- Comparison Models
- Real machine execution time

Context: Which Optimizations ?

Any optimization

- Only consider sequences of elementary optimizations
- One optimization module:
  - A computable function that terminates
  - Can have any number of parameters
  - Includes preliminary analysis, if needed
- If it does not apply, does not perform any transformation

Optimizations are blackboxes:

- Optimizations mapped to letters, sequences to words;
- Infinite number of sequences;
- Possible infinite number of optimized programs.

Phase Ordering Problem Formulation

We assume the compiler knows:

- A performance model t;
- A set of optimization modules $\mathcal{M}$;
- The performance bound to reach.

Phase Ordering Problem Formulation

We assume the compiler knows:

- A performance model t;
- A set of optimization modules $\mathcal{M}$;
- The performance bound to reach.

Assumptions on the compiler and machine:

- Each optimization $m \in \mathcal{M}$ optimizes a program, independently of the input;
- No optimization parameters;
- Sequence of optimizations: $s = m_0 \dots m_n$
- Performance evaluation depends on the optimized program and on the input: $t(s\mathcal{P}, I)$.

Phase Ordering Problem Formulation 1

Problem formulation

Given t, T and $\mathcal{M}$, is there an algorithm that determines for each program and input a sequence $s \in \mathcal{M}^*$ such that

$$t(s(\mathcal{P}), I) < T?$$

Phase Ordering Problem Formulation 1

Problem formulation

Given t, T and $\mathcal{M}$, is there an algorithm that determines for each program and input a sequence $s \in \mathcal{M}^*$ such that

$$t(s(\mathcal{P}), I) < T?$$

Depends on t:

- For some values of t, the problem is simple (simplified machine)
- For some other values (real machine), the problem is difficult

Phase Ordering Problem Formulation 2

t given by the real machine:

- The compiler does not really know the real performance model;
- t is any function that checks the partial machine description of compiler;
- There exists a program with an infinite number of optimized versions.

Phase Ordering Problem Formulation 2

t given by the real machine:

- The compiler does not really know the real performance model;
- t is any function that checks the partial machine description of compiler;
- There exists a program with an infinite number of optimized versions.

Given and $\mathcal{M}$,

Problem formulation

Given t and $\mathcal{M}$, is there an algorithm that determines for each program and input a sequence $s \in \mathcal{M}^*$ such that

$$t(s(\mathcal{P}), I) < T?$$

Phase Ordering Problem Formulation 2

t given by the real machine:

- The compiler does not really know the real performance model;
- t is any function that checks the partial machine description of compiler;
- There exists a program with an infinite number of optimized versions.

Given and $\mathcal{M}$,

Problem formulation

Given t and $\mathcal{M}$, is there an algorithm that determines for each program and input a sequence $s \in \mathcal{M}^*$ such that

$$t(s(\mathcal{P}), I) < T?$$

UNDECIDABLE

Phase Ordering Problem for Library Generation

We assume:

- The compiler does not know the real performance model;
- The compiler knows the program and input;
- The compiler knows the bound to reach;
- The program has an infinite number of optimized versions.

Phase Ordering Problem for Library Generation

We assume:

- The compiler does not know the real performance model;
- The compiler knows the program and input;
- The compiler knows the bound to reach;
- The program has an infinite number of optimized versions.

Given and $\mathcal{M}, \mathcal{P}, I, T$,

Problem formulation

Given and $\mathcal{M}, \mathcal{P}, I, T$, is there an algorithm that determines for each real machine t a sequence $s \in \mathcal{M}^*$ such that

$$t(s(\mathcal{P}), I) < T?$$

Phase Ordering Problem for Library Generation

We assume:

- The compiler does not know the real performance model;
- The compiler knows the program and input;
- The compiler knows the bound to reach;
- The program has an infinite number of optimized versions.

Given and $\mathcal{M}, \mathcal{P}, I, T$,

Problem formulation

Given and $\mathcal{M}, \mathcal{P}, I, T$, is there an algorithm that determines for each real machine t a sequence $s \in \mathcal{M}^*$ such that

$$t(s(\mathcal{P}), I) < T?$$

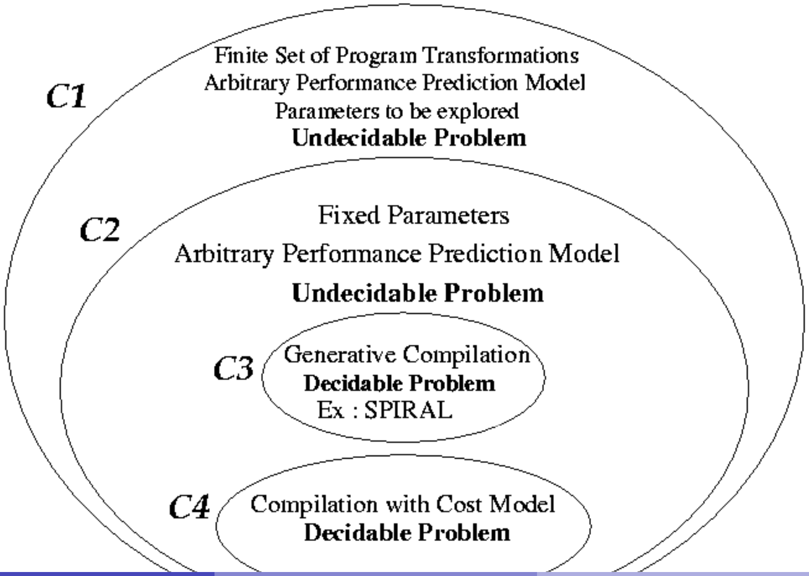
UNDECIDABLE

Phase Ordering: Decidable Cases

Limiting the number of sequences:

- Limiting the compilation cost (or number of optimizations);
- One pass generative compilers [Spiral,FFTW]

Phase Ordering



Parameter Space Exploration

Parameter space exploration corresponds to some library generation (e.g. ATLAS).
We assume:

- The sequence s of optimizations is given;
- Each optimization has a given number of parameters (unroll degree,...): an optimization m applied to $\mathcal{P}$ with parameters k, $m(\mathcal{P}, k)$.
- Parameters are not bounded
- Performance evaluation function t is known by the compiler

Parameter Space Exploration Problem 1

Given sequence s and function t:

Problem Formulation

Is there an algorithm that finds for all program $\mathcal{P}$, input I and bound T the parameters k:

$$t(s(\mathcal{P}, k), I) < T$$

Parameter Space Exploration Problem 1

Given sequence s and function t:

Problem Formulation

Is there an algorithm that finds for all program $\mathcal{P}$, input I and bound T the parameters k:

$$t(s(\mathcal{P}, k), I) < T$$

Solution: depends on t and s...

Parameter Space Exploration Problem 2

The performance evaluation function is built in two steps:

- A function is built according to s and the program: $t(s, \mathcal{P})$
- The performance is evaluated according to I and k: $t(s, \mathcal{P})(k, I)$
- $t(s, \mathcal{P})$ is any polynomial
  - Corresponds to possible program complexity metrics

Parameter Space Exploration Problem 2

The performance evaluation function is built in two steps:

- A function is built according to s and the program: $t(s, \mathcal{P})$
- The performance is evaluated according to I and k: $t(s, \mathcal{P})(k, I)$
- $t(s, \mathcal{P})$ is any polynomial
  - Corresponds to possible program complexity metrics

Given sequence s and function t:

Problem Formulation

Is there an algorithm that finds for all program $\mathcal{P}$, input I and bound T the parameters k:

$$t(s, \mathcal{P})(k, I) < T$$

Parameter Space Exploration Problem 2

The performance evaluation function is built in two steps:

- A function is built according to s and the program: $t(s, \mathcal{P})$
- The performance is evaluated according to I and k: $t(s, \mathcal{P})(k, I)$
- $t(s, \mathcal{P})$ is any polynomial
  - Corresponds to possible program complexity metrics

Given sequence s and function t:

Problem Formulation

Is there an algorithm that finds for all program $\mathcal{P}$, input I and bound T the parameters k:

$$t(s, \mathcal{P})(k, I) < T$$

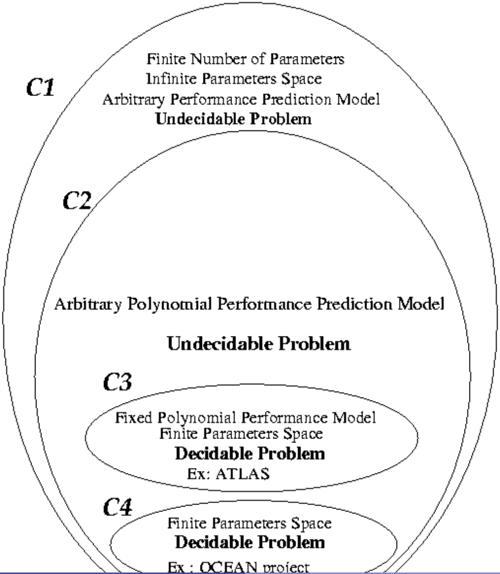
UNDECIDABLE

Parameter Exploration: Decidable Cases

Limit the parameter space to be finite:

- OCEAN project: parameters in finite intervals
- Atlas: parameters bounded according to a model (cache size)

Parameter Exploration



Conclusion

More decidable cases:

- Optimizations considered as blackboxes. What happens for a certain class of optimizations ?
  - Finite number of optimized codes ? (no parameters)
  - Sum up of multiple optimizations ? (as for unimodular transformations)
- Which performance model for which optimizations ?
- Limit parameter space, optimization combination through user input, application knowledge ?
- Have an exact performance prediction model of real machines

Thank You !