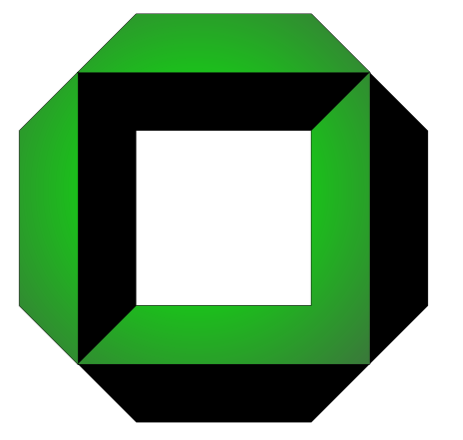




Optimal Coalescing

Daniel Grund*
Saarland University

Sebastian Hack
University of Karlsruhe

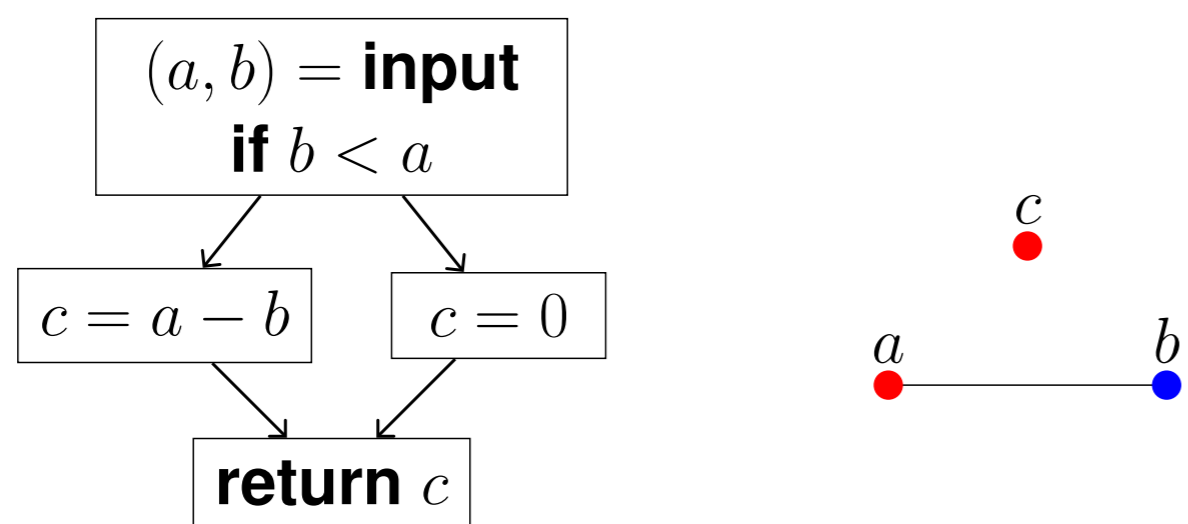


Introduction

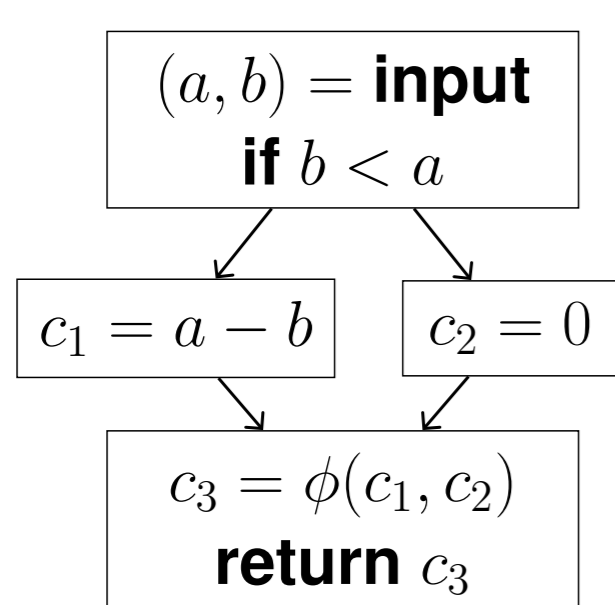
An important step in compiling a program to machine code is the allocation of CPU registers for the program's variables. Coalescing is a subtask of register allocation which removes unnecessary copies from the program. This reduces the code size and increases performance. Large parts of the problem can be reduced to graph coloring.

Interference Graphs - IFG

- Variables interfere if they are simultaneously live
- For each variable there is a node in the graph
- Nodes of interfering variables are adjacent
- Coloring gives register allocation



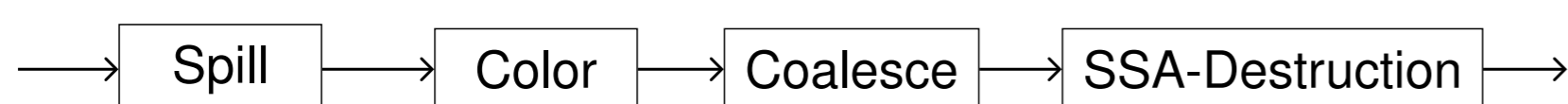
Static Single Assignment – SSA



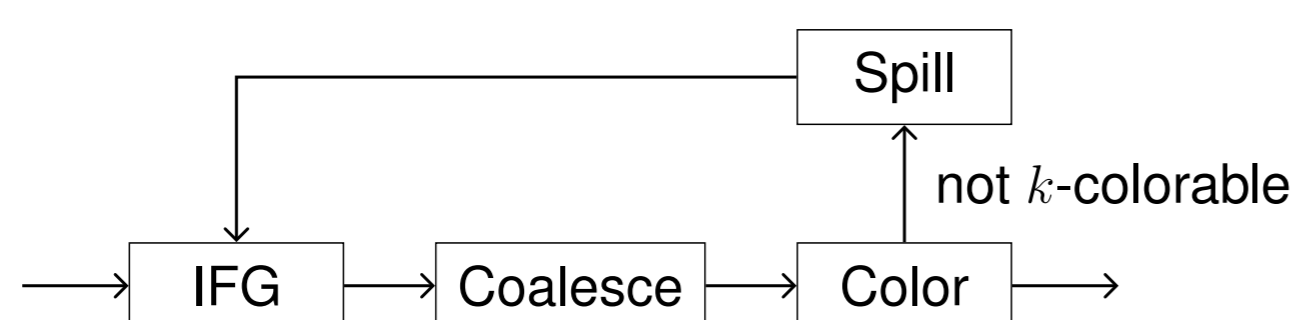
- Each variable has exactly one definition
- Abstract ϕ -operations select values dependent on control flow

The Trigger

Recent results: Interference graphs of programs in SSA form are special - they are *chordal*. Chordal graphs have nice properties which allow for a “linear” register allocation like

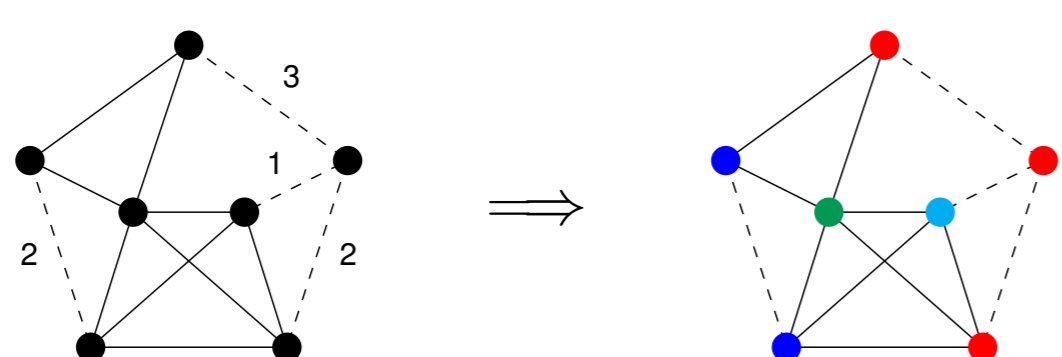


in contrast to the former “iterative” allocators.



The Problem

In this new setting the coalescing problem can be represented as a chordal graph augmented with weighted *affinity edges*. Such an edge connects source and destination of a copy and represents its runtime or code-size costs.

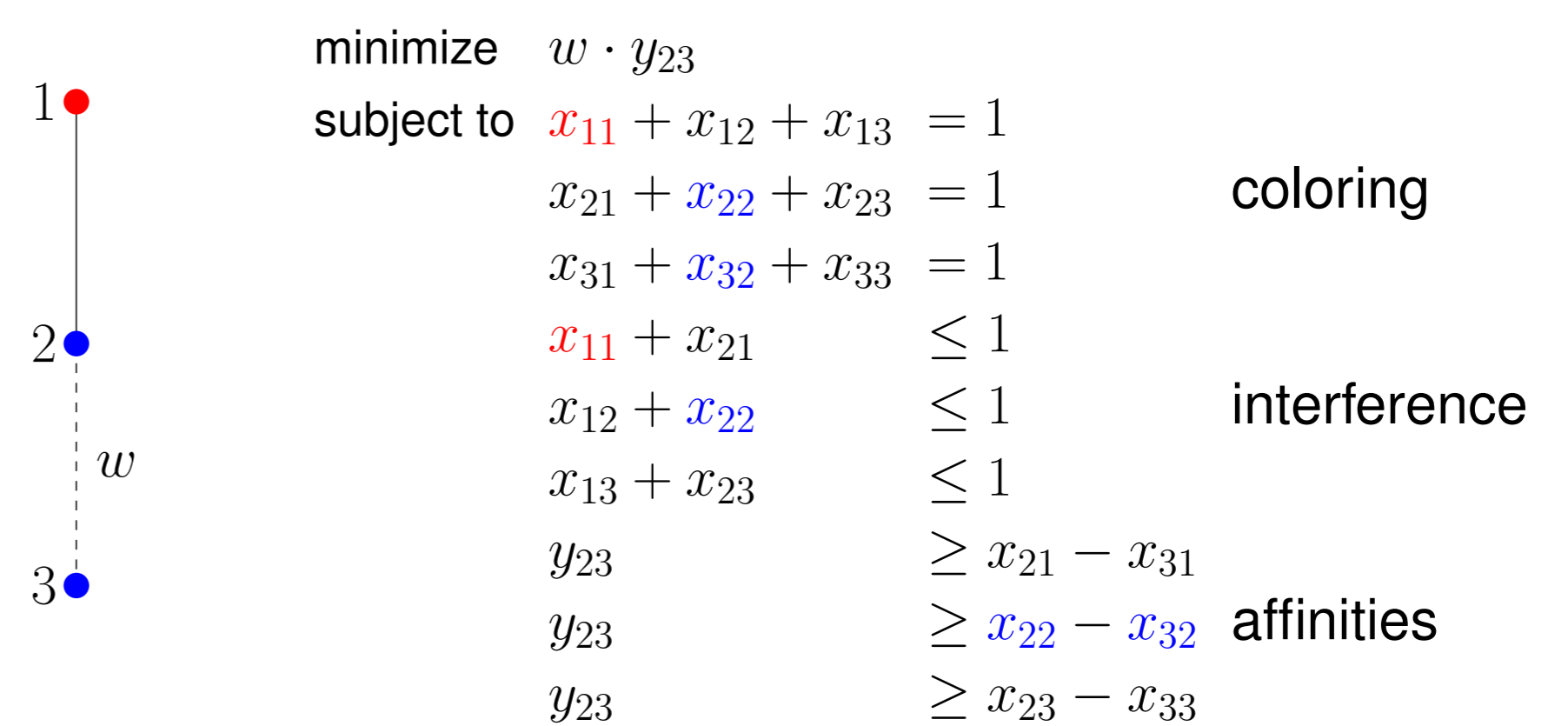


Solution

We reduce this \mathcal{NP} -complete problem to Integer Linear Programming (ILP). ILP is a general method solving the problem of maximizing or minimizing an objective function subject to (in)equality constraints and integrality restrictions on a (sub)set of variables.

Basic ILP Model

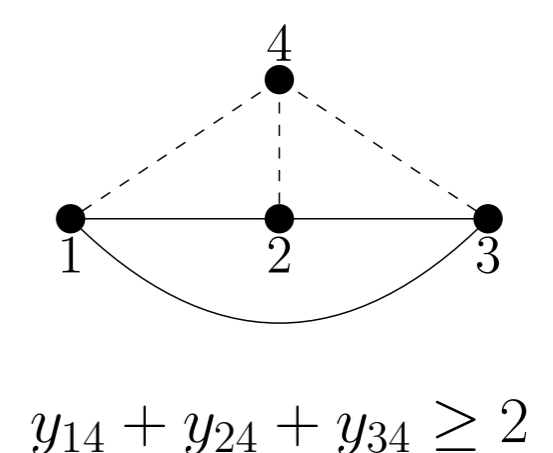
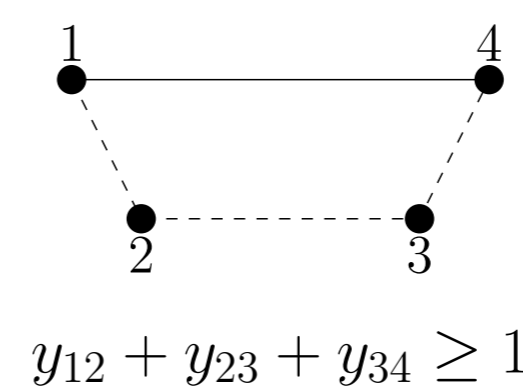
- Binary variables represent states/decisions
- Coloring: $x_{ic} = 1 \iff$ node i has color c
- Optimality: $y_{ij} = 1 \iff$ node i and j have different colors



Optimizations

To increase the performance of the solving process we apply various optimizations:

- Exploiting chordality reduces problem size
- Chordality allows efficient encoding of interferences
- Cutting planes reduce search space, e.g. a path cut (left) and a clique ray cut (right)



Results

Conclusions after benchmarking these algorithms:

- Solving efficient due to chordality and optimizations
- Faster than all previous ILP approaches
- First practical optimal algorithm

Reference: Daniel Grund and Sebastian Hack: *A Fast Cutting-Plane Algorithm for Optimal Coalescing*. In Compiler Construction - CC 2007, LNCS 4420, Springer EAPLS Best Paper Award 2007

Contact information:

<http://rw4.cs.uni-sb.de/>
grund@cs.uni-sb.de

*Partially supported by the

Deutsche
Forschungsgemeinschaft

