Minimising overall jumps in CPS-style compilers

TIANYU LI
(supervisor Dan Ghica)

1

Background and motivation

- CPS-style compilation uses a lot of jumping at the level of assembly
- Non-local jumps are expensive
  - Normal flow of control is cheapest
  - Local jumps are more expensive
  - Non-local jumps are very expensive (cache misses approx. 200 cycles)
- Ghica's *Geometry of Synthesis* compiler can compile to CPS-style "flat" (not procedures) code

Minimise jumping costs!

abstract

1. Mathematic model
   dynamic programming – analysis

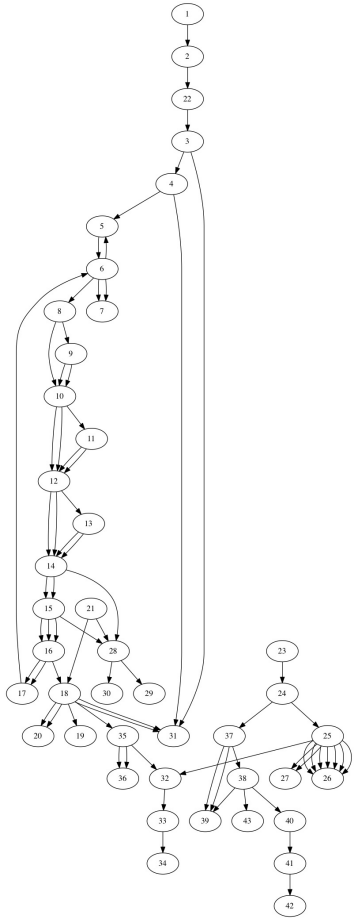
2. Definition of implicit links
   generalized cost function to evaluate jumps
   maximising the number of implicit links

3. Be described as N traveling salesman problem (TSP)
   was proof as NP-complete problem
   different heuristics

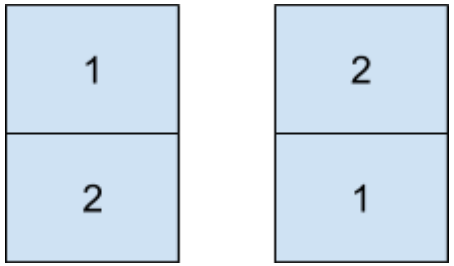
4. Implementation and evaluation
   benchmarks …

Assembly code

Abstract representation of assembly code



Cost models



Analysis - dynamic programming

Dynamic programming method is efficient method to optimise permutation problem.

Dynamic programming (also known as dynamic optimisation) is a method for solving a complex problem by breaking it down into a collection of simpler sub-problems.

Same strategy – might be a NP complete problem

Approximate dynamic programming:

implicit link

1.the link between adjacency blocks

2.the amount of line less than K

Equivalence to TSP

If the first block of code is decided

Heuristic approach

Nearest neighbor:

select the link with maximum amount of implicit link in each iteration

Insertion:

insert a new node in the place maximum the amount of implicit link in each iteration

To do

1. Comparison and evaluation (more testing)
   complexity …

2. Improve efficiency
   Binary state compression - dp
   Minimising data using topological sorting (greedy)
   Sorting before insertion

3. Other data representation (attempt) – sparse matrix

3. More literature (from other area)
   Prefer solution

4. Optional
   Implementation in Ocaml

THANKS