Instruction Selection
Instruction Selection on SSA

- “Optimal” instruction selection on trees is polynomial
- SSA programs are directed graphs
  \[ \Rightarrow \] Data dependence graphs
- Translating back from SSA graphs to trees is not satisfactory
- “Optimal” instruction selection is NP-complete on DAGs
- The problem is common subexpressions
- Doing it on graphs provides more opportunities for complex instructions:
  - Patterns with multiple results
  - DAG-like patterns
Instruction Selection on SSA

- Graph Rewriting
- For every machine instruction specify:
  - A set of graphs (patterns) of IR nodes
  - Every pattern has associated costs

1. Find all matchings of the patterns in the IR graph
2. Pick a correct and optimal matching
3. Replace each pattern by corresponding machine instruction

⇒ Result is an SSA graph with machine nodes
Graphs

- Let $G = (V, E)$ be a directed acyclic graph (DAG)
- Let $Op$ be a set of operators
- Every node has a degree $\deg v : V \rightarrow \mathbb{N}_0$
- Every node $v \in V$ has an operator: $\text{op} : V \rightarrow Op$
- Every operator $o \in Op$ has an arity $\# : Op \rightarrow \mathbb{N}_0$
- Let $\square \in Op$ be an operator with $\# \square = 0$
- Nodes with operator $\square$ denote "glue" points in the patterns (later)
- Every node's degree must match the operator's arity:
  \[ \# \text{op } v = \deg v \]

Definition (Program Graph)

A graph $G$ is a program graph if it is acyclic and

\[ \forall v \in V : \text{op } v \neq \square \]
A graph $P = (V, E)$ is **rooted** if there exists a node $v \in V_P$ such that there is a path from $v$ to every node $v'$ in $P$.

If $P$ is rooted, denote the root by $rt\ P$.

**Definition (Pattern Graph, Pattern)**

A graph $P$ is a pattern if

- it is acyclic and rooted
- $\text{op}\ rt\ P \neq \square$

- Note that we explicitly allow nodes with operator $\square$ in patterns.
Equivalence of Nodes in Patterns

- Complex patterns often have common sub-patterns

- Shall be treated as equivalent

- Selecting the common sub-pattern at the Add node shall enable selecting the complex instruction at Store and Load
Equivalence of Nodes in Patterns

Definition (Equivalence of nodes)

Consider two patterns $P$ and $Q$ and two nodes $v \in P$, $w \in Q$:

\[ v \sim w \iff v = w \land (\text{span } v \cong \text{span } w \land \text{rt } P \neq v \land \text{rt } Q \neq w) \]

- Either the two nodes are identical
- $v$, $w$ are no pattern roots and their spanned subgraphs are isomorphic
- $\text{span } v$: induced subgraph that contains all nodes reachable from $v$
Matching of a Node

- Let $\mathcal{P} = \{P_1, P_2, \ldots \}$ be a set of patterns
- Let $G$ be some program graph

Definition (Matching)

A matching $\mathcal{M}_v$ of a node $v \in V_G$ with a set of patterns $\mathcal{P}$ is a family of pairs

$$\mathcal{M}_v = \{(P_i, \iota_i)\}_{i \in I}, \quad I \subseteq \{1, \ldots, |\mathcal{P}|\}$$

of patterns and injective graph morphisms $\iota_i : P_i \to G$ satisfying

$$v \in \text{ran} \iota_i \quad \text{and} \quad \text{op } w \neq \Box \implies \text{op } w = \text{op } \iota_i(w) \quad \forall w \in P_i$$
Matchings

Example

Pattern $P_A$

Program Graph

Pattern $P_B$
We have computed a covering of the graph

i.e. instruction selection possibilities

Now, find a subset of the covering that leads to good and correct code

Cast the problem as a mathematical optimization problem:

Partitioned Boolean Quadratic Programming (PBQP)
PBQP

Let $\mathbb{R}_\infty = \mathbb{R}_+ \cup \{\infty\}$ and

- $\vec{c}_i \in \mathbb{R}^{k_i}_\infty$ be cost vectors
- $C_{ij} \in \mathbb{R}^{k_i}_\infty \times \mathbb{R}^{k_j}_\infty$ be cost matrices

Definition (PBQP)

Minimize

$$\sum_{1 \leq i < j \leq n} \vec{x}_i^\top \cdot C_{ij} \cdot \vec{x}_j + \sum_{1 \leq i \leq n} \vec{x}_i^\top \cdot \vec{c}_i$$

with respect to

- $\vec{x}_i \in \{0, 1\}^{k_i}$
- $\vec{x}_i^\top \cdot \vec{1} = 1, \quad 1 \leq i \leq n$
- $\vec{x}_i^\top \cdot C_{ij} \cdot \vec{x}_j < \infty, \quad 1 \leq i < j \leq n$
\( \tilde{x}_i \) are selection vectors

Exactly one component is 1

This selects the component

Cost matrices relate selection of made in different selection vectors

Can be modelled as a graph:

- cost vectors are nodes
- matrices are edges
- only draw non-null matrix edges
PBQP as a Graph

Colors indicate selection vectors $\vec{x}_i = (0 \ 1 \ 0)^\top$ and $\vec{x}_j = (1 \ 0)^\top$

This selection contributes the cost of 6 to the global costs

Edge direction solely to indicate order of $ij$ in the matrix subscript
Mapping Instruction Selection to PBQP

Add

Add + Const

Const

\[
\begin{align*}
\text{Add} & \quad \text{Add} \\
\text{Add} & \quad \text{Add} \\
\text{Const} & \quad \text{Const} \\
\end{align*}
\]
Mapping Instruction Selection to PBQP

Cost vectors are defined by node coverings:

- Let $\mathcal{M}_v$ be a node matching of $v$

- The alternatives of the node are given by partitioning the matchings by equivalence:
  
  $\mathcal{M}_v / \sim$

- Common sub-patterns have to result in the same choice

- Costs come from an external specification
Mapping Instruction Selection to PBQP

- Matrices have to maintain selection correctness

- Consider two alternatives

\[ A_u = (P_u, \iota_u) \quad A_v = (P_v, \iota_v) \]

at two nodes \( u, v \) connected by an edge \( u \rightarrow v \).

- The matrix entry for those alternatives is

\[
c(A_u, A_v) = \begin{cases} 
\infty & \text{op} \iota_u^{-1}(v) = \Box \text{ and } \iota_v^{-1}(v) \neq rt P_v \\
\infty & \text{op} \iota_u^{-1}(v) \neq \Box \text{ and } \iota_u^{-1}(v) \not\sim \iota_v^{-1}(v) \\
0 & \text{else}
\end{cases}
\]

Id est:

- If \( A_u \) selects a leaf at \( v \), \( A_v \) has to select a root

- If \( A_u \) does not select a leaf, both subpatters have to be equivalent
Example
Program Graph

```
Example
Program Graph

```
Example

Patterns

LAC (Load + Add + Const)  LA (Load + Add)  AC (Add + Const)

Const  Phi  Load  Add

C (Const)  P (Phi)  L (Load)  A (Add)
Example

Matchings

\[ \text{Load} \]

\[ \text{Add} \]

\[ \text{Const} \]

\[ C, AC, LAC_1, LAC_2 \]

\[ \text{P} \]

\[ \text{Phi} \]

\[ A, AC, LA_1, LAC_1, LA_2, LAC_2 \]

\[ L_1, LA_1, LAC_1 \]

\[ L_2, LA_2, LAC_2 \]
Example
PBQP Instance
Reducing the Problem

Optimality-preserving reductions of the problem:

- Independent edges (e.g. matrix of zeroes):

  ![Diagram of independent edges]

- Nodes of degree 1:

  ![Diagram of nodes of degree 1]

- Nodes of degree 2:

  ![Diagram of nodes of degree 2]
Reducing the Problem

- Heuristic Reduction:
  
  Chose the local minimum at a node

- Leads to a linear algorithm

- Each reduction eliminates at least one edge

- If all edges are reduced, minimizing nodes separately is easy
Summary

- Map instruction selection to an optimization problem
- SSA graphs are sparse \(\Rightarrow\) reductions often applied
- In practice: heuristic reduction rarely happens
- Efficiently solvable
- Convenient mechanism:
  - Implementor specifies patterns and costs
  - maps each pattern to an machine node
  - Rest is automatic

- Criteria for pattern sets that allow for correct selections in every program not discussed here!
Literature

