Register Loading via Linear Programming

Gruia Calinescu Minming Li

May 28, 2014

Abstract

We study the following optimization problem. The input is a number k and a directed graph with a specified "start" vertex, each of whose vertices may have one "memory bank requirement", an integer. There are k "registers", labeled $1 \dots k$. A valid solution associates to the vertices with no bank requirement one or more "load instructions" $L[b, j]$, for bank b and register j, such that every directed trail from the start vertex to some vertex with bank requirement c contains a vertex u that has been associated $L[c, i]$ (for some register $i \leq k$) and no vertex following u in the trail has been associated an $L[b, i]$, for any other bank b. The objective is to minimize the total number of associated load instructions.

We give a $k(k+1)$-approximation algorithm based on linear programming rounding, with $(k+1)$ being the best possible unless Vertex Cover has approximation $2-\epsilon$ for $\epsilon > 0$. We also present a $O(k \log n)$ approximation, with n being the number of vertices in the input directed graph. Based on the same linear program, another rounding method outputs a valid solution with objective at most $2k$ times the optimum for k registers, using $2k-1$ registers.

This version of the paper corrects some minor errors that made it in the final Algorithmica paper.

1 Introduction

Partitioned memory architecture is common in 8-bit microcontrollers. For example, Freescale [1] 68HC11 8-bit microcontrollers allow multiple 64KB memory banks to be accessed by their 16-bit address registers with only one bank being active at a time. Zilog [2] Z80 also addresses a maximum of 64 KB memory using 16-bit address registers. Other examples include Intel 8051 processor family and MOS technology 6502 series microcontrollers. For embedded systems using these 8-bit microcontrollers, how to insert bank selection instructions (or load instructions) to minimize the code size is an important research topic. Given a Control Flow Graph, where each node represents some code block, we can insert bank selection instructions (loading a bank index to a register) either immediately before the block or immediately after the block to activate some memory bank. We want to use the minimum number of bank selection instructions to guarantee that no matter from which path the program enters the current code block, the memory bank which contains this block is active at this moment. Formal definitions are in the next subsection.

Note that we are not optimizing the run-time of the program (in which case the problem would resemble caching [26]) nor the number of registers needed for a program (as in Thorup [28], Kannan and Proebsting [19], and Jansen and Reiter [18]). An early related work is [16]. More recent work appears in [25], while [13, 20, 21, 27, 22] and many other papers deal with practical

issues of NP-hard variants of register allocation. Also, as opposed to most theoretical work, we do not assume any structure (such as low treewidth) for the input graph. Most related to our model are the "spill heuristics" discussed in [6, 5, 7, 29, 12, 10], but as the name suggests we do not know of any previous approximation algorithms. Here "spill" means putting some variables in the RAM instead of registers and the aim of those heuristics is to minimize the number of variables "spilled".

So, while our problem resembles register allocation, it differs in the following ways. We do not force a bank with a "live-range" to stay in the same register but allow the register to change content over time. We also have the restriction that a bank variable cannot be stored in RAM when it is to be visited (it must come back to a register in time, which is different from the Register Allocation problem where a variable can be spilled). In the new setting, our goal is to minimize the total number of content switching instructions for registers inserted into the program.

We organize the remaining of the paper as follows. Subsection 1.1 gives the original problem formulation k-OBSIM together with the more pure version k-BSIM. Subsection 1.2 describes previous and new results and further discussion. In Section 2, we show a reduction to prove the hardness of k-OBSIM and also show that the k-OBSIM problem can be transformed to the k-BSIM problem and hence we will focus on solving k-BSIM. Section 3 presents our integer linear program for 1-BSIM and the rounding procedure giving the 2-approximation. We also discuss derandomization in Subsection 3.2, using linear programming duality to reduce the running time of the rounding procedure. Section 4 presents the generalization of the approximation results from 1-BSIM to k-BSIM, as well as the two other approximation results mentioned in the abstract. We conclude in Section 5.

1.1 Problem Formulation

Starting directly from embedded systems, we obtain the Original k-Bank Selection Instruction Minimization problem (k-OBSIM), defined as follows: the input is a number k and a directed graph, called the Control Flow Graph (CFG), with a specified "start" vertex, and for each vertex we have at most one "memory bank requirement", an integer. The vertices of the CFG correspond to blocks of code in an embedded system, and arcs represent possible jumps in the code. Many embedded systems use partitioned memory architecture, and program variables are stored in "banks" that must be activated by storing its index in registers before use. For illustration convenience, we simply say a bank is stored in a register when the bank is activated. A vertex with no bank requirement is called a *transparent* vertex, and a vertex with one bank requirement is called a *required* vertex. There are k registers, labeled $1 \dots k$. Let B be the set of possible banks.

Each vertex of the CFG may "load" a bank (use a bank load instruction), either at the "entrance" or at the "exit" of the vertex (or both). We write $L^u_{in}[b, j]$ for loading bank b in register j at the entrance of vertex u, and $L^u_{out}[b, j]$ for loading at the exit. Although loading is also allowed on the arcs of the CFG, we prefer to subdivide such arcs with transparent vertices to keep the problem description simpler. Also, when $k = 1$, a small proof shows that any load on an arc can be done instead at the entry of the head of the arc, resulting in another feasible solution not worse in the number of load instructions.

For a trail (directed path, not necessarily simple) P, let $\widehat{P}$ denote the set of interior vertices of P, obtained as follows: from the sequence of vertices of P, remove the first and last vertex,

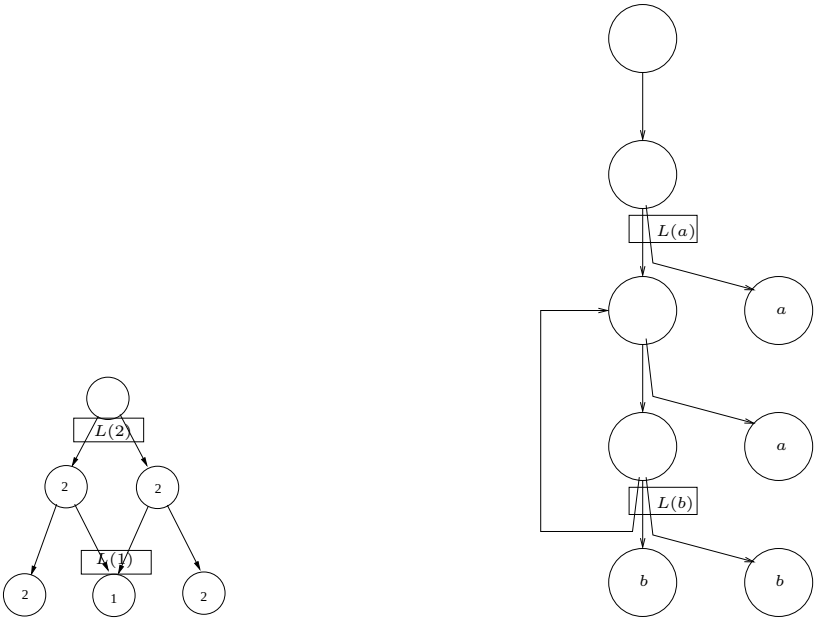

Figure 1: On the *left*: An example input with a feasible solution when $k = 1$. *Circles* represent nodes and the number in the circle means the bank required by this node (*empty circles* are transparent nodes). The start vertex is not represented, or it could be the top (transparent) vertex. With only one register, and the position w.r.t. a vertex clear from the picture, we write $L(a)$ to mean "load bank a". On the *right*: another example (also $k = 1$) with an unfeasible solution. There is a non-simple path going to the lower vertex with bank requirement a that loads a, then b, and does not load a again.

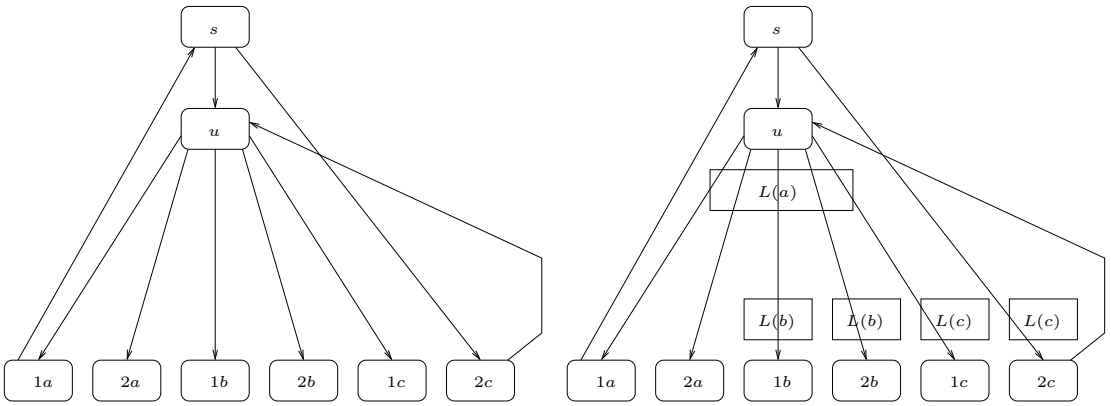

Figure 2: On the *left*, an instance with directed cycles, for $k = 1$. Nodes $1a$ and $2a$ require bank a, nodes $1b$ and $2b$ require bank b, nodes $1c$ and $2c$ require bank c, while the start s and node u are transparent. On the *right*, a feasible solution. We abbreviate $L_{out}[b, j]$ or $L_{in}[b, j]$ to $L(b)$ since the figure shows the position of the load instructions, and there is exactly one register.

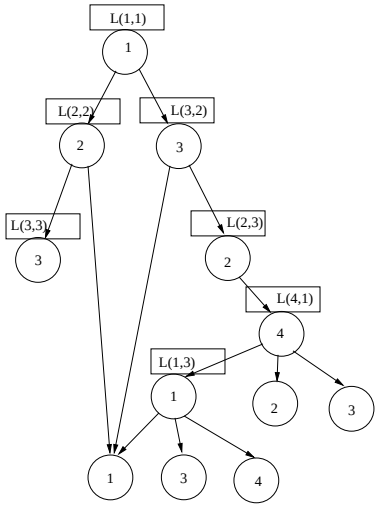

Figure 3: An example input with a feasible solution when $k = 3$. For the bottom node which accesses bank 1, there are three paths entering it, where two of them have bank 1 loaded in register 1 and one of them has bank 1 loaded in register 3. We abbreviate $L_{out}[b, j]$ or $L_{in}(b, j)$ to $L(b, j)$ since the figure shows the position of the load instructions. The start vertex is not represented and has only one arc, to the top-most vertex.

and then eliminate duplicates. Note that the start or end of P may appear in $\widehat{P}$. If for trail P we have vertex $w \in \widehat{P}$, let $P[w]$ be the subtrail of P from the last occurrence of w in P to the end vertex of P. Let s be the start vertex of the CFG. In a feasible solution, bank load instructions must be associated to CFG vertices such that, for any trail P from s to some node v that has a bank requirement b, P has a vertex w (which may be v) and a register j for some $j \leq k$ such that one of the following holds:

1. $w = v$, and we have $L^w_{in}[b, j]$ but no $L^w_{in}[c, j]$ for any $c \in B$ with $c \neq b$
2. w has $L^w_{out}[b, j]$, and for no $c \in B$ with $c \neq b$ there is $L^v_{in}[c, j]$ or $L^w_{out}[c, j]$ or a vertex u of $\widehat{P[w]}$ with either $L^u_{out}[c, j]$ or $L^u_{in}[c, j]$
3. w has $L^w_{in}[b, j]$, and for no $c \in B$ with $c \neq b$ there is $L^v_{in}[c, j]$ or $L^w_{in}[c, j]$ or $L^w_{out}[c, j]$ or a vertex u of $\widehat{P[w]}$ with either $L^u_{out}[c, j]$ or $L^u_{in}[c, j]$.

We call such a trail *fulfilled*. See figures 1, 2, and 3 for examples of feasible solutions. In other words, in a trail from the start vertex to some vertex v requiring bank b, bank b is always loaded in some register and there are no other bank loads over b further on the trail. The objective is to minimize the total number of bank load instructions (as to keep the embedded code as short as possible). See the right example in Figure 1 for a scenario where simple paths can be fulfilled but there exists an unfulfilled trail.

We prefer to work with a slightly more pure problem, called k-BSIM. The input is a number k and a directed graph with a specified "start" vertex, each of whose vertices may have one "memory bank requirement", an integer. There are k "registers", labeled $1 \ldots k$. A valid solution associates to the vertices with no bank requirement one or more "load instructions" $L[b, j]$, for

bank b and register j, such that every directed trail from the start vertex to some vertex with bank requirement c contains a vertex u that has been associated $L[c, i]$ (for some register $i \leq k$) and no vertex following u in the trail has been associated an $L[b, i]$, for any other bank b. The objective is to minimize the total number of associated load instructions.

The difference between k-OBSIM and k-BSIM is that in k-BSIM, "load instructions" can only be added inside nodes with no bank requirement (transparent nodes) while in k-OBSIM, "load instructions" can be added before and after any node.

1.2 Results and Discussion

The paper [23] studies 1-OBSIM without transparent nodes, showing NP-hardness and a 2-approximation algorithm. We generalize this result, by giving a $k(k+1)$-approximation algorithm for k-OBSIM (transparent nodes also allowed) based on linear programming rounding. In a personal communication, Yuan Zhou [31] claimed that 1-OBSIM without transparent nodes does not have a $2 - \epsilon$ approximation algorithm unless Vertex Cover has a $2 - \epsilon$ approximation algorithm (and it is believed that such an algorithm does not exist). We also present such a reduction, and generalize it to show that k-OBSIM without transparent nodes does not have a $\alpha - \epsilon$ approximation algorithm unless $(k + 1)$-uniform Hypergraph Vertex Cover has such an algorithm. It is known that it is NP-hard to approximate Hypergraph Vertex Cover in a r-uniform-hypergraph to within a factor of $(r - 1 - \epsilon)$ [11], and it is believed that an approximation ratio of r is the best a polynomial-time algorithm can do. Thus it is NP-hard to approximate k-OBSIM ($k > 2$) within a factor of $k - \epsilon$.

The results in [23] only work for acyclic CFGs with no transparent nodes, while all our approximation algorithms work for arbitrary CFGs (with cycles and transparent nodes) by adding an essential new constraint and also using a more sophisticated rounding analysis, which is akin to how Node-Multiway-Cut [14] generalizes Vertex Cover. For $k = 1$, the existence of transparent nodes poses a serious challenge, since given a transparent node v, solutions without a "load instruction" at v may exist, and for such a solution, different program flows (trails) going through v can have different banks being active when leaving v. However, for a node with a bank requirement, no matter which flow the program goes through, the active bank will be the same after going through this node. For $k > 1$, there are similar issues.

Based on the same linear program, we also present a $O(k \log n)$-approximation, with n being the number of vertices in the input directed graph, for k-BSIM. Another rounding method outputs a valid solution with objective at most $2k$ times the optimum for k registers, however using $2k-1$ registers instead of k registers. The linear program contains one "clever" constraint which makes it similar, for $k = 1$, to the linear program used by Garg, Vazirani, and Yannakakis [14] to obtain a 2-approximation for Node Weighted Multiway Cut. We discussed earlier the hardness of k-BSIM, obtained from the hardness of k-uniform Hypergraph Vertex Cover. Informally, k-BSIM also inherits some hardness from k-Coloring (as in the register allocation papers [28] and [19], with the latter using, as one of our algorithms above, $2k$ instead of k registers) and we see intuitive connections to Directed Steiner Tree [30, 8, 32] and Multicut in Directed Graphs [9, 15, 3].

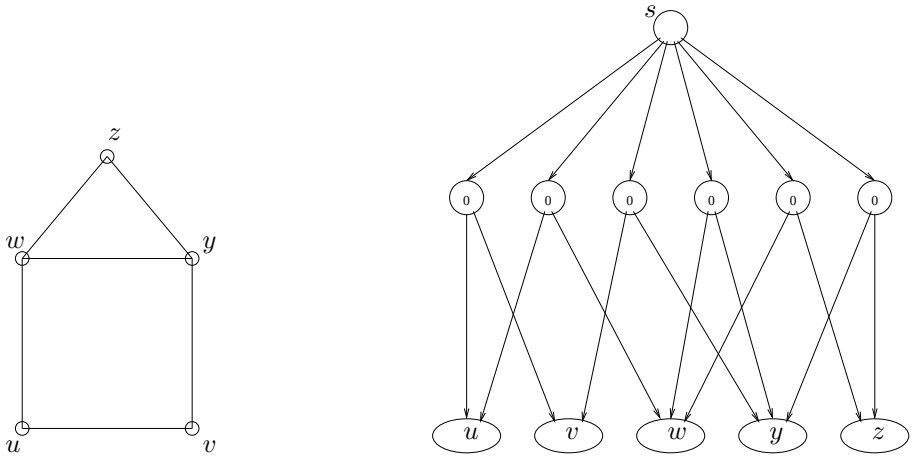

Figure 4: An example of reducing Vertex Cover (instance on the *left*) to 1-OBSIM (instance on the *right*; each *oval* contains many vertices with the same bank requirement).

2 Reductions

A reduction from Vertex Cover to 1-OBSIM without transparent nodes was announced by Yuan Zhou [31]. We show directly how $(k+1)$-uniform Hypergraph Vertex Cover reduces to k-OBSIM without transparent nodes.

r-uniform Hypergraph Vertex Cover is the following problem: The input $H = (V, E)$ is a r-uniform hypergraph; that is each hyperedge $e \in E$ is a subset of V of size r. A set C of vertices is said to cover a hyperedge e if $e \cap C \neq \emptyset$, and is said to be a vertex cover if it covers all hyperedges. The objective is to find a minimum size vertex cover. Vertex Cover is 2-uniform Hypergraph Vertex Cover.

Please refer to Figure 4 for an illustration. Given a $(k+1)$-uniform Hypergraph Vertex Cover instance $G = (V, E)$ (with $|V| = n$ and $|E| = m$), construct the CFG as follows: add to its vertices many (say, km^2) copies of each vertex of $v \in V$, all with required bank v, creating *groups* with vertices in different groups having different requirements. For each hyperedge $e \in E$, add to the CFG a vertex with bank $0 \notin V$, and put an arc from this vertex to all the copies of the vertices of V included in e. Add to the CFG the start vertex s and put arcs from it to all the CFG-vertices obtained from hyperedges of E. Call this k-OBSIM instance I.

We claim that if G has a vertex cover C of size q, we can find a feasible solution for I that uses at most $1 + km + qkm^2$ bank load instructions: (please refer to Figure 5 for an illustration) all the qkm^2 copies of all the vertices in C will load their respective requirement at the entry, all the vertices of I obtained from some $e \in E$ will load, at the exit, the requirement of the vertices in $e \setminus C$ (there are at most k such vertices since e has cardinality $(k+1)$ and at least one vertex of C is contained in the set e). s loads bank 0 at the exit.

Moreover, if I has a solution with less than qkm^2 bank load instructions, then G has a vertex cover Q of size less than q: put in Q a vertex v if all the km^2 copies of v load their requirement. Then $|Q| < q$ and it remains to prove that all the hyperedges of G are covered by Q. Indeed, if hyperedge $e \in E$ is not covered by Q, then for every $u \in e$, there is a copy of u with no bank load instruction; however then no matter what k load instructions we associate with the vertex

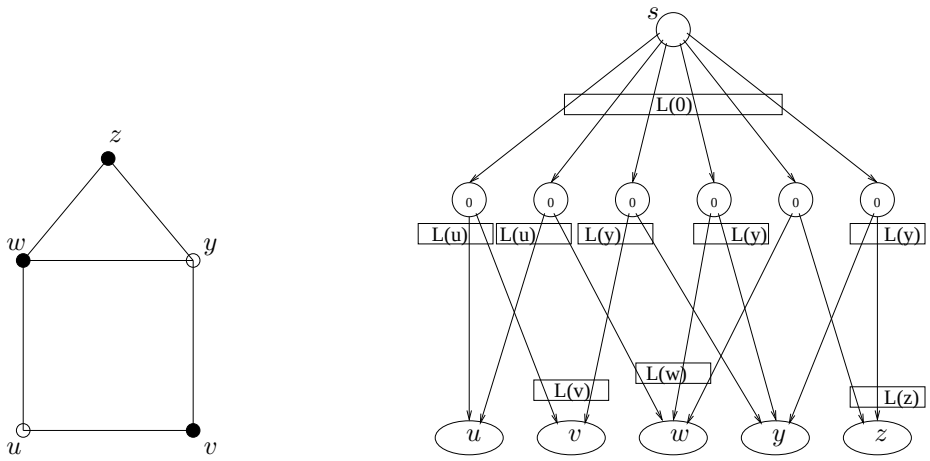

Figure 5: A feasible solution for the Vertex Cover instance of Figure 4 appears on the *left*. The corresponding 1-OBSIM feasible solution appears on the *right*; as each *oval* contains many vertices with the same bank requirement, there will be many $L(v)$, $L(w)$, and $L(y)$ instructions.

of I corresponding to e or with the start vertex, we do not obtain a feasible solution for I: since $|e| > k$ there will be a vertex $w \in e$ such that bank w is not loaded after leaving the vertex of I corresponding to e, and since also one of w's copy (call it w') in I does not have an associated load instruction at its entry, the trail (here, a simple path) s, e, w' is not fulfilled.

In conclusion, if G has optimum vertex cover of size q, I has optimum between qkm^2 and $qkm^2 + km + 1$, and thus k-OBSIM cannot be approximated with ratio $k - \epsilon$ unless $P = NP$ (using [11]) and 1-OBSIM cannot be approximated with ratio $2 - \epsilon$ unless Vertex Cover can be approximated with ratio $2 - \epsilon$.

We continue by showing how k-OBSIM reduces to k-BSIM (with transparent nodes), a problem easier to describe. Given an instance of k-OBSIM, for every node v in CFG with bank requirement b, add a transparent node v^t_{in} which takes in all the incoming arcs of v and has one arc to v, thus v has exactly one incoming arc. Also add a transparent node v^t_{out} which sends out all the outgoing arcs of v, and has one arc from v, thus v has exactly one outgoing arc. If for the k-OBSIM instance, there is a load operation at the entrance of some vertex v with bank requirement b, then in the transformed k-BSIM instance, we do the same load to node v^t_{in}; if there is a load operation at the exit of some vertex v with bank requirement b, then in the transformed k-BSIM instance, we do the same load to node v^t_{out}. Thus, with the above correspondence, a feasible solution for the k-OBSIM instance can be changed to a feasible solution for the transformed k-BSIM instance. Also, it is easy to see that a feasible solution for the transformed k-BSIM instance can be changed to a feasible solution for the original k-OBSIM instance, without an increase in the objective function.

3 1-BSIM

Remove nodes from the CFG such that every node is reachable from the start vertex s. We do a transformation for the given OBSIM instance, similar to the previous reduction. The linear

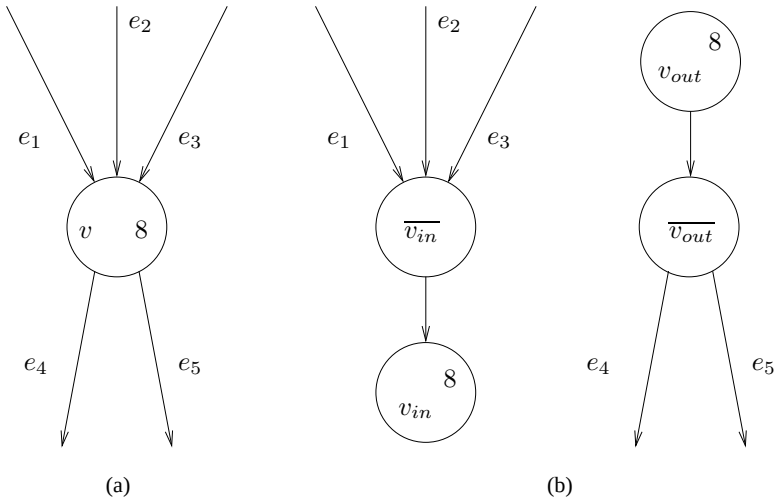

Figure 6: Splitting a node as described in the reduction from 1-OBSIM to BSIM. From node v with bank requirement 8 (a), four nodes are created (b). $\overline{v_{in}}$ is the transparent node taking in all the incoming arcs of v_{in}, and $\overline{v_{out}}$ is the transparent node sending out all the incoming arcs of v_{out}.

program obtained later after this transformation is more intuitive (but this reduction only works for $k = 1$).

Create a new transparent start vertex, s', with exactly one arc, outgoing to the original s. For every node v in CFG with bank requirement b, split v in two nodes, v_{in} with all the incoming arcs of v, and v_{out} with all the outgoing arcs of v; both have requirement b. This operation is illustrated in Figure 6. Note that we do not have an arc from v_{in} to v_{out}. Moreover, for v_{in}, add a transparent node $\overline{v_{in}}$ which takes in all the incoming arcs of v_{in} and has one arc to v_{in}, thus v_{in} has exactly one incoming arc. Also, for v_{out}, add a transparent node $\overline{v_{out}}$ which sends out all the outgoing arcs of v_{out}, and has one arc from v_{out}, thus v_{out} has exactly one outgoing arc. We now insist that all load instructions are done at transparent nodes. Figure 7 gives an example.

Call the resulting directed graph $G = (V, E)$. Let $R^I \subset V$ (lowest level in Figure 7) contain all the required nodes with one incoming arc each (that is, the v_{in} nodes), let $R^O \subset V$ (highest level in Figure 7) contain s' and the required nodes with one outgoing arc each (that is, the v_{out} nodes), and let F be the set of transparent nodes other than s' (in Figure 7, these nodes are neither on the highest nor lowest levels). For $a \in B$, let R^I_a be the subset of R^I with requirement a, and R^O_a be the subset of R^O with requirement a. In G, we insist that for every bank $a \in B$ and every vertex $v \in R^I_a$, every trail ending in v and starting at a vertex of $R^O \setminus R^O_a$ contains a vertex $u \in F$ loading bank a, and no load instructions after u. Call BSIM this new problem. One can check that a 1-OBSIM feasible solution for the original instance corresponds to a BSIM feasible solution to the constructed instance, with the same number of load instructions.

For $v \in V$ and $a \in B$, let $\mathcal{T}^v_a$ be the (possibly infinite) set of trails of G from v to some node of R^I_a, and let $\mathcal{P}^v_a$ be the set of simple paths of G from v to some node of R^I_a. Write the following integer linear program (**IP1**), with variables x^v_b for every node $v \in F$ and bank requirement $b \in B$ (x^v_b in the IP would be 1 if node $v \in F$ loads bank b), and variables d^v_b for every node $v \in \left(F \cup R^O\right)$

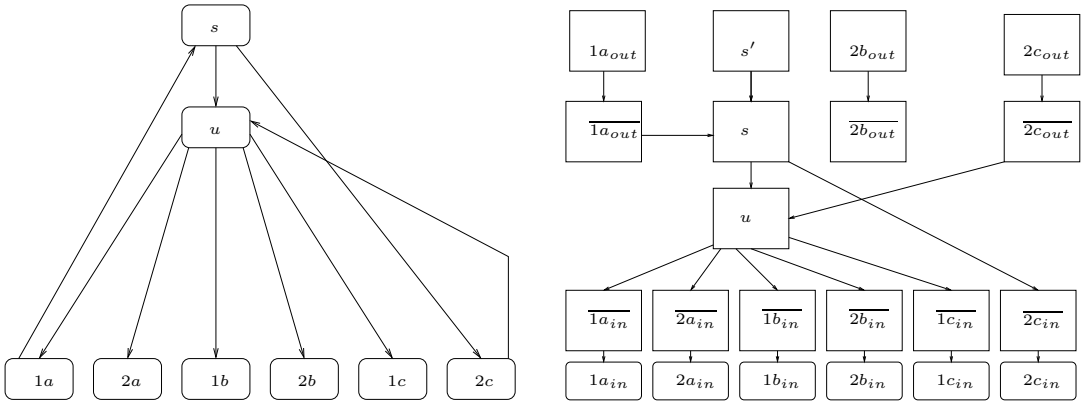

Figure 7: On the left, the instance from Figure 2. Nodes $1a$ and $2a$ require bank a, nodes $1b$ and $2b$ require bank b, nodes $1c$ and $2c$ require bank c, while the start s and node u are transparent. On the right, the BSIM instance created (not represented are three weakly connected components with $2a_{out}$, $1b_{out}$, and $1c_{out}$). Note also that weakly connected component of $2b_{out}$ plays no role in finding optimal solutions.

and bank requirement $b \in B$ (d_b^v in the IP would be 1 if either $\mathcal{P}_b^v = \emptyset$ or, for any simple path $P \in \mathcal{P}_b^v$, $\widehat{P}$ contains at least one node that loads bank b). Note that $\mathcal{P}_b^v = \emptyset$ iff $\mathcal{T}_b^v = \emptyset$, and that if any simple path $P \in \mathcal{P}_b^v$, $\widehat{P}$ contains at least one node that loads bank b, then for any trail $T \in \mathcal{T}_b^v$, $\widehat{T}$ contains at least one node that loads bank b. Also note that if, for some $v \in \left(F \cup R^O\right)$, we have that for any simple path $P \in \mathcal{P}_b^v$, $\widehat{P}$ contains at least one node that loads bank b, it does not necessarily follow that bank b "arrives loaded at the destination" on such a path since it is not mentioned that another bank is not loaded "over b" later on the path.

$$\min \sum_{v \in F, b \in B} x_b^v$$

$$\text{subject to } \sum_{b \in B} x_b^v \leq 1 \qquad \forall v \in F \tag{1}$$

$$d_a^u \geq 1 \qquad \forall a \in B \wedge \forall u \in \left(R^O \setminus R_a^O\right) \tag{2}$$

$$d_a^v \geq x_b^v \qquad \forall a \neq b \in B \wedge \forall v \in F \tag{3}$$

$$d_a^u \leq d_a^v + x_a^v \qquad \forall a \in B \wedge \forall u \in \left(F \cup R^O\right) \wedge \forall v \in F \text{ such that } uv \in E \tag{4}$$

$$d_a^u = 0 \qquad \forall a \in B \wedge \forall u \in F \text{ such that } \exists v \in R_a^I \text{ such that } uv \in E \tag{5}$$

$$d_a^u + d_b^u \geq 1 \qquad \forall a \neq b \in B \wedge \forall u \in F \tag{6}$$

$$x_a^v \geq 0 \qquad \forall v \in F \wedge \forall a \in B \tag{7}$$

$$d_a^v \geq 0 \qquad \forall v \in \left(F \cup R^O\right) \wedge \forall a \in B \tag{8}$$

$$x_a^v, d_a^v \in \mathbb{Z} \qquad \forall v \in V \wedge \forall a \in B \tag{9}$$

See an example in Figure 8. We argue the fact that any IP solution obtained from a BSIM solution satisfies all these constraints, and that we can construct a valid BSIM solution from any IP solution. It is rather obvious the objective function of the two feasible solutions matches.

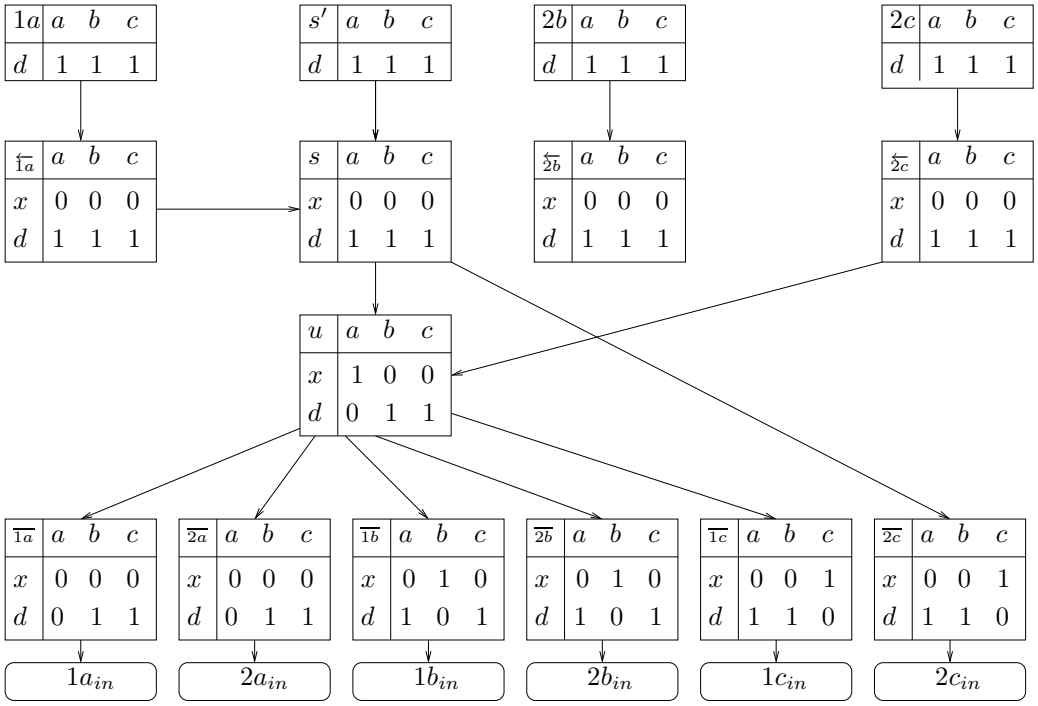

Figure 8: A feasible solution to the integer program instance from the BSIM instance from Figure 7. Nodes $1a$ and $2a$ require bank a, nodes $1b$ and $2b$ require bank b, nodes $1c$ and $2c$ require bank c, while the original start s and node u are transparent. To fit the notations in the figure, $1a_{out}$ is replaced by $1a$, $\overline{1a_{out}}$ is $1a$ with an arrow on top, and $\overline{1a_{in}}$ is replaced by $\overline{1a}$. This **IP1** feasible solution corresponds to the 1-OBSIM feasible solution from Figure 2.

Constraint (1) enforces only one load per vertex of F. It holds for the same reason. Constraint (2) enforces the condition that for every bank $a \in B$ and every vertex $v \in R_a^I$, every simple path (and also every trail) ending in v and starting at a vertex of $R^O \setminus R_a^O$ contains a transparent vertex loading bank a; it does not guarantee however no load instructions after u. This is done by Constraint (3), which enforces the following observation: if bank b is loaded in vertex v, then for any simple path (and also every trail) from v to a vertex requiring bank a, there must be at least one load of bank a. These are also the reasons Constraints (2) and (3) hold for an IP solution cosntructed from a BSIM solutions as explained before the description of **IP1**.

Constraint (4) holds for the the following reason: if for bank a and vertices u, v with $uv \in E$, we have $\mathcal{P}_a^v \neq \emptyset$ and there exists a simple path $P \in \mathcal{P}_a^v$ such that $\widehat{P}$ contains no node that loads bank a, and v also does not load a, then $\mathcal{P}_a^u \neq \emptyset$ and there exists a path $P' \in \mathcal{P}_a^u$ (namely, shortcut if needed the trail starting with arc uv followed by P) such that $\widehat{P'}$ contains no node that loads bank a. Constraint (5) holds since, if $v \in R_a^I$ and $uv \in E$, then $\mathcal{P}_a^u \neq \emptyset$ and there exists a trail $P' \in \mathcal{P}_a^u$ (namely, arc uv) such that $\widehat{P'}$ contains no node that loads bank a.

The trickier to verify constraint is (6), which indeed holds for integer solutions as, if for vertex v and banks $a \neq b$, $\mathcal{P}_a^v$ and $\mathcal{P}_b^v$ are both non-empty, then no matter if or what bank is loaded in v or in any other free vertex, either we must have that every simple path $P \in \mathcal{P}_a^v$ satisfies that $\widehat{P}$ contains at least one node that loads bank a, or we must have that every simple path $P \in \mathcal{P}_b^v$ satisfies that $\widehat{P}$ contains at least one node that loads bank b. Indeed, if there is a simple path $P \in \mathcal{P}_a^v$ with $\widehat{P}$ not loading a, then we must have that either v loads a, or all the simple paths from R^O to v load a or are coming from R_a^O (and a trail from R^O to v must exist since we assumed every vertex of the CFG is reachable from s). Thus if such a P exists, we must have that every simple path $P' \in \mathcal{P}_b^v$ satisfies that $\widehat{P'}$ contains at least one node that loads bank b. It is the crucial (and clever) Constraint (6) that allows good approximation algorithms.

Now, given an IP1 feasible solution, it remains to argue that loading bank b at vertex u whenever $x_b^u = 1$ gives a feasible BSIM solution. Indeed, let $a \in B$, $v \in R_a^I$ and P be a trail from some $w \in \left(R^O \setminus R_a^O\right)$ to v. Constraints (2),(4), and (5) ensure that at least one vertex u of $\widehat{P}$ has $x_b^u = 1$. Pick z to be the last such vertex of $\widehat{P}$. If any vertex y following z on $\widehat{P}$ has $x_a^y = 1$ then Constraint (3) ensures $d_b^y = 1$ and therefore another vertex v of $\widehat{P}$, following y, has $x_b^v = 1$, contradicting the selection of z.

3.1 LP rounding

Let **LP1** be the linear programming relaxation of **IP1**, which can be solved in polynomial time. See Figure 9 for an example of a fractional solution. Let $\bar{x}_a^v, \bar{d}_a^v$ be an optimum **LP1** solution. Pick uniformly at random a real number $\delta \in (0, 1/2)$. Set (for all possible v, a) $x_a^v = 1$ iff $\bar{d}_a^v < \delta \leq \bar{d}_a^v + \bar{x}_a^v$. Set (for all possible v, a) $d_a^v = 1$ iff $\mathcal{P}_a^v = \emptyset$ or any path P in $\mathcal{P}_a^v$ has some $u \in \widehat{P}$ with $x_a^u = 1$ (this can be achieved by Breadth First Search). It is immediate that $Pr[x_a^v = 1] \leq 2\bar{x}_a^v$, and thus we have a 2-approximation, provided we prove that for any such δ, we get a valid **IP1** solution.

Lemma 3.1 *For any $\delta \in (0, 1/2)$, and for any $v \in (F \cup R^O)$ and $b \in B$, if $\bar{d}_b^v \geq 1/2$ and $\mathcal{P}_b^v \neq \emptyset$, then any simple path $P \in \mathcal{P}_b^v$ has a vertex $z \neq v$ with $x_b^z = 1$ (in the BSIM instance G, vertex z load bank b).*

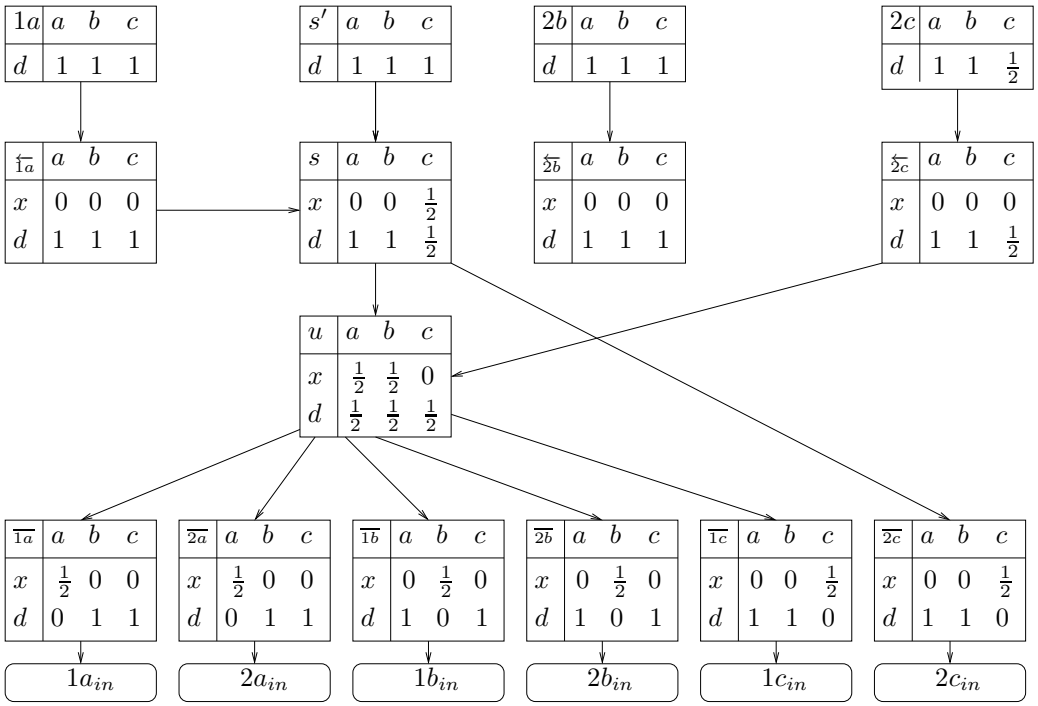

Figure 9: A feasible solution to the linear program instance from the BSIM instance from Figure 7. Nodes $1a$ and $2a$ require bank a, nodes $1b$ and $2b$ require bank b, nodes $1c$ and $2c$ require bank c, while the original start s and node u are transparent. To fit the notations in the figure, $1a_{out}$ is replaced by $1a$, $\overline{1a_{out}}$ is $1a$ with an arrow on top, and $\overline{1a_{in}}$ is replaced by $\overline{1a}$. This **LP1** feasible solution has objective 4.5, while one can verify that the 1-OBSIM instance from Figure 2 only has solutions with objective 5 or more.

Proof: Let P be such a simple path from v to some $w \in R_b^I$. By Constraint (5), the vertex y before w in P has $\bar{d}_b^y = 0$. Therefore P must have consecutive vertices u and u' such that $\bar{d}_b^{u'} < \delta$ and $\bar{d}_b^u \geq \delta$; here u may be v. Note that $u' \in F$. Constraint (4) also gives $\bar{d}_b^{u'} + \bar{x}_b^{u'} \geq \bar{d}_b^u \geq \delta$, and therefore $x_b^{u'}$ is set to 1 by the algorithm. The lemma holds with $z = u'$. □

Now we check the feasibility of all constraints. For Constraint (1), note that for $a \neq b \in B$ and $v \in F$, in order to have both x_a^v and x_b^v be made 1, we must have $\bar{d}_a^v < 1/2$ and $\bar{d}_b^v < 1/2$, leading to $\bar{d}$ violating Constraint (6).

Constraint (2), for $a \in B$ and $u \in R^O \setminus R_a^O$ follows from $\bar{d}_a^u \geq 1$ and the lemma above (if $\mathcal{P}_a^u \neq \emptyset$), or the way we set all $d_a^v = 1$ above if $\mathcal{P}_a^v = \emptyset$.

Constraint (3), for $a \neq b \in B$ and $v \in F$ follows from the following argument: if $x_b^v = 1$, then $\bar{d}_b^v < 1/2$, and therefore by $\bar{d}$ satisfying Constraint (6), $\bar{d}_a^v \geq 1/2$. Therefore, by the lemma above applied to v and a, we set $d_a^v = 1$ whether $\mathcal{P}_a^v = \emptyset$ or not.

Constraint (4), for $a \in B$ and $uv \in E$, follows from the way d was constructed: if both $d_a^v = 0$ and $x_a^v = 0$, then $\mathcal{P}_a^u \neq \emptyset$ since $\mathcal{P}_a^v \neq \emptyset$, and there is a simple path $P \in \mathcal{P}_a^u$ such that, for all $z \in \widehat{P}$, $x_a^z = 0$: obtain P by shortcutting, if needed, the trail that starts with arc uv and finishes with a simple path $P' \in \mathcal{P}_a^v$ with, for all $z \in \widehat{P'}$, $x_a^z = 0$. The existence of P implies $d_a^u = 0$. If $d_a^v \neq 0$ or $x_a^v \neq 0$, then $d_a^v = 1$ or $x_a^v = 1$, so Constraint (4) is satisfied.

Constraint (5), for $a \in B$ and $u \in F$ such that there exists $uv \in E$ with $v \in R_a^I$ is also satisfied since $\mathcal{P}_a^u \neq \emptyset$ and the simple path with its only arc uv has no interior.

Constraint (6), for $a \neq b \in B$ and $u \in F$ follows as follows: either $\bar{d}_a^u \geq 1/2$ or $\bar{d}_b^u \geq 1/2$, and the lemma above ensures that the one at least 1/2 becomes 1. Constraints (7), (8), and (9) are immediate.

3.2 Derandomization

Note that only a polynomial number of values of δ must be tried, so derandomization is immediate. We go further, and write a relaxation of **LP1**, and its dual and use complementary slackness to show that every value of δ gives a 2-approximation, as it also happens for the linear program of Garg, Vazirani, and Yannakakis [14]. Let **LP1'** be the variant of **LP1** without constraints (1) and (3); **LP1'** is a relaxation of BSIM and thus any BSIM feasible solution within 2 of the optimum of **LP1'** is a 2-approximation. Note that we do not claim that the integral version of **LP1'** is equivalent to the original BSIM instance.

The following linear program with exponentially many constraints (as we only consider simple paths) can be seen to be equivalent to (and solved by) **LP1'**.

$$\min \sum_{v \in F, b \in B} x_b^v$$

$$\text{subject to } \sum_{v \in \widehat{P}} x_a^v \geq 1 \qquad \forall a \in B \wedge \forall u \in \left(R^O \setminus R_a^O\right) \wedge \forall P \in \mathcal{P}_a^u \tag{10}$$

$$\sum_{v \in \widehat{P}_1} x_a^v + \sum_{v \in \widehat{P}_2} x_b^v \geq 1 \qquad \forall a < b \in B \wedge \forall u \in F \wedge \forall P_1 \in \mathcal{P}_a^u \wedge \forall P_2 \in \mathcal{P}_b^u \tag{11}$$

$$x_a^v \geq 0 \qquad \forall v \in F \wedge a \in B \tag{12}$$

To see the equivalence, use the same values x_a^v, and set in **LP1'**: $d_a^u = \min_{P \in \mathcal{P}_a^u} \sum_{v \in \widehat{P}} x_a^v$.

The program above is in fact a covering linear program and combinatorial $1+\epsilon$ approximations also exist [24, 17]; however the method below requires an optimum, and one would need to try all δ if only an approximate linear programming optimum is given.

The dual of the above program, given below, has variables α_P for all $a \in B$ and for all $u \in \left(R^O \setminus R_a^O\right)$ and for all $P \in \mathcal{P}_a^u$, and variables β_{P_1,P_2} (the order of the paths matter) for all $a < b \in B$, all $u \in F$, all $P_1 \in \mathcal{P}_a^u$ and all $P_2 \in \mathcal{P}_b^u$.

$$\max \sum_{a \in B} \sum_{u \in (R^O \setminus R_a^O)} \sum_{P \in \mathcal{P}_a^u} \alpha_P + \sum_{a \in B} \sum_{b \neq a \in B} \sum_{u \in F} \sum_{P_1 \in \mathcal{P}_a^u} \sum_{P_2 \in \mathcal{P}_b^u} \beta_{P_1,P_2}$$

$$\text{subject to} \sum_{u \in (R^O \setminus R_a^O)} \sum_{P \in \mathcal{P}_a^u \mid v \in \widehat{P}} \alpha_P + \sum_{b < a \in B} \sum_{u \in F} \sum_{P_2 \in \mathcal{P}_a^u \mid v \in \widehat{P}_2} \sum_{P_1 \in \mathcal{P}_b^u} \beta_{P_1,P_2} \tag{13}$$

$$+ \sum_{b > a \in B} \sum_{u \in F} \sum_{P_1 \in \mathcal{P}_a^u \mid v \in \widehat{P}_1} \sum_{P_2 \in \mathcal{P}_b^u} \beta_{P_1,P_2} \leq 1 \ \ \forall v \in F \wedge \forall a \in B \tag{14}$$

$$\alpha_P \geq 0 \tag{15}$$

$$\beta_{P_1,P_2} \geq 0 \tag{16}$$

Let $\bar{x}_a^v, \bar{d}_a^v$ be an optimum solution to the primal. Pick any real number $\delta \in (0, 1/2)$. Set $\hat{x}_a^v = 1$ iff $\bar{d}_a^v < \delta \leq \bar{d}_a^v + \bar{x}_a^v$.

Claim 3.1 *$\hat{x}$ gives a valid BSIM solution.*

Proof: Assume $\hat{x}_a^v = 1$. This implies $\bar{d}_a^v < 1/2$, and Constraint (11) gives that for all $b \neq a \in B$, $\bar{d}_b^v > 1/2$, and thus $\hat{x}_b^v = 0$.

Note also that Lemma 3.1 holds as well (same proof). Let P be an arbitrary trail from a vertex of $R^O \setminus R_b^O$ to a vertex of R_b^I. Let P' be the simple path obtained by short-cutting P. Then Constraint (10) ensures the existence of a vertex u in $\widehat{P}'$ with $\hat{x}_b^u = 1$. Then such a vertex also exists on P, and choose w to be the last vertex on P with $\hat{x}_b^w = 1$. Thus on trail P, w loads bank b.

Suppose for a contradiction that some vertex v that follows or equals w on P loads bank $a \neq b$. This means that $\bar{d}_a^v < 1/2$ and Constraint (11) ensures that $\bar{d}_b^v \geq 1/2$. Let P'' be the simple path from v to the endpoint of P obtained by short-cutting P. Lemma 3.1 gives a vertex $z \neq v$ on $\widehat{P}''$ with $\hat{x}_b^z = 1$; note that $\widehat{P}$" is a subtrail of P that strictly follows w, contradicting the choice of w. Thus no vertex v that follows or equals w on P loads bank $a \neq b$, which means that P is fulfilled. As P was arbitrary, the claim follows. □

As for the approximation ratio of 2, write the complementary slackness conditions (below, in the summation $\sum_{v \in \widehat{P}} x_a^v$, the bank a is such that path P ends at a vertex of R_a^I) :

$$\alpha_P > 0 \implies \sum_{v \in \widehat{P}} x_a^v = 1 \tag{17}$$

$$\beta_{P_1,P_2} > 0 \implies \sum_{v \in \widehat{P}_1} x_a^v + \sum_{v \in \widehat{P}_2} x_b^v = 1 \tag{18}$$

$$x_a^v > 0 \implies \sum_{\ldots} \alpha_P + \sum_{\ldots} \beta_{P_1,P_2} = 1 \tag{19}$$

which hold for $\bar{x}$ and an optimum dual solution (Condition (19) says Constraint (14) is tight; we did not give all the details above). With respect to the same dual solution, it is immediate that $\hat{x}_a^v > 0$ only if $\bar{x}_a^v > 0$ and therefore Condition (19) holds. Any path $P \in \mathcal{P}_a^u$ with $\alpha_P > 0$ must have that going through the vertices $v \in \widehat{P}$, we see non-increasing $\bar{d}_a^v$-values (we know P is a shortest path w.r.t $\bar{x}_a$, from Condition (17)), and thus only one such vertex v can have $\hat{x}_a^v = 1$, and thus Condition (17) is respected by $\hat{x}$ as well. Any two paths P_1, P_2 with $\beta_{P_1,P_2} > 0$ must be such that P_1 and P_2 are each a shortest path, and thus as argued above, $\widehat{P}_1$ has at most one v with $\hat{x}_a^v = 1$ and $\widehat{P}_2$ has at most one w with $\hat{x}_b^w = 1$. For $\hat{x}$, Condition (18) holds approximately - with a factor of 2, and as in the primal-dual method, we obtain that $\hat{x}$ is a 2-approximation of $\bar{x}$.

We do not see half-integrality as in [14], and we do not see a direct primal-dual algorithm.

4 k-BSIM

Without loss of generality, assume that every node is reachable from the start vertex s. For technical reasons, add a new transparent start vertex, s', with exactly one arc, outgoing to the original s (s is transparent as well, or else the instance does not have valid solutions). Let F be the set of transparent nodes other than s', R be the set of required vertices, and, for $a \in B$, let R_a be the subset of R with requirement a. As before, for $v \in V$ and $a \in B$, $\mathcal{T}_a^v$ denotes the (possibly infinite) set of trails of G from v to some node of R_a^I, and $\mathcal{P}_a^v$ denotes the set of simple paths of G from v to some node of R_a^I.

Write the following integer linear program (**IP2**), with variables x_b^v for every node $v \in F$ and bank requirement $b \in B$ (x_b^v in the IP would be 1 if transparent node v loads bank b, in any of its registers), and variables d_b^v for every bank requirement $b \in B$ and node $v \in ((F \cup R \cup \{s'\}) \setminus R_b)$ (d_b^v in the IP would be 1 if either $\mathcal{P}_b^v = \emptyset$ or, for any $P \in \mathcal{P}_b^v$, $\widehat{P}$ contains at least one node that loads bank b, in any register).

$$\min \sum_{v \in F, b \in B} x_b^v$$

$$\text{subject to } \sum_{b \in B} x_b^v \leq k \qquad \forall v \in F \tag{20}$$

$$d_a^{s'} \geq 1 \qquad \forall a \in B \tag{21}$$

$$\sum_{a \in B} d_a^u \geq |B| - k \qquad \forall u \in F \tag{22}$$

$$d_a^u \leq d_a^v + x_a^v \qquad \forall a \in B \wedge \forall u \in (F \cup R \setminus R_a) \wedge \forall v \in F \text{ such that } uv \in E \tag{23}$$

$$d_a^u \leq d_a^v \qquad \forall a \in B \wedge \forall u \in (F \cup R \setminus R_a) \wedge \forall v \in (R \setminus R_a) \text{ such that } uv \in E \tag{24}$$

$$d_a^u = 0 \qquad \forall a \in B \wedge \forall u \in (F \cup R \setminus R_a) \text{ such that } uv \in E \tag{25}$$

$$1 \geq x_a^v \geq 0 \qquad \forall v \in F \wedge \forall a \in B \tag{26}$$

$$1 \geq d_a^v \geq 0 \qquad \forall a \in B \wedge \forall v \in (F \cup R \setminus R_a) \tag{27}$$

$$x_a^v, d_a^v \in \mathbb{Z} \qquad \forall v \in F \wedge \forall a \in B \tag{28}$$

Constraints (22) are the generalization of the "clever" constraints (6); in fact they could (and should, if one uses an IP solver) be used together with Constraints (27) to replace Constraints

(6) in **LP1**. Indeed, Constraints (22) hold for integer solutions since, if for transparent vertex v there are $k+1$ banks b with variable $d_b^v = 0$, then for any of these banks b, $\mathcal{P}_b^v \neq \emptyset$ and there is at least one path $P \in \mathcal{P}_b^v$ such that no node in $\widehat{P}$ loads bank b in any of its registers. Then no matter what banks arrive or are loaded at v (which is reachable from s'), we do not get a valid k-BSIM solution. Note that constraints

$$\sum_{a \in Q} d_a^u \geq |Q| - k \;\; \forall Q \subseteq B \wedge \forall u \in F \tag{29}$$

are implied by (27) and (22).

IP2 above is not equivalent to k-BSIM, as it does not specify in which register a bank is loaded. Nevertheless, from a k-BSIM solution, we can get an **IP2** solution of the same value (but not vice versa; that will be a coloring problem), by setting x_b^v to be 1 iff transparent node v loads bank b and d_b^v to be 1 iff either $\mathcal{P}_b^v = \emptyset$ or, for any $P \in \mathcal{P}_b^v$, $\widehat{P}$ contains at least one node that loads bank b. We relax **IP2** to the linear program **LP2**, and solve it in polynomial time.

Let $\bar{x}_a^v, \bar{d}_a^v$ be an optimum **LP2** solution. Pick uniformly at random a real number $\delta \in (0, 1/(k+1))$. Set $x_a^v = 1$ iff $\bar{d}_a^v < \delta \leq \bar{d}_a^v + \bar{x}_a^v$.

It is immediate that $Pr[x_a^v = 1] \leq (k+1)\bar{x}_a^v$. We load at v all the q banks a with $\bar{d}_a^u < 1/(k+1)$ if at least one of them has $x_a^v = 1$, using registers $1, 2, \ldots, q$. We have $q \leq k$, since Constraint (29) implies that, for any vertex u, at most k banks a can have $\bar{d}_a^u < 1/(k+1)$. Thus the expected number of loads is at most $k(k+1)$ times the LP cost.

One needs to check that indeed this is a valid solution of k-BSIM: pick an arbitrary $a \in B$, an arbitrary $u \in R_a$, and an arbitrary trail P from s' to u. Let u' be the last vertex of P in F; u' exists since s is on P and in F. From Constraints (25) and (24) we get $\bar{d}_a^{u'} = 0$. From Constraints (21) we get $\bar{d}_a^{s'} \geq 1/k$. Let v be, as we go on P, the last vertex (last occurrence also) such that $\bar{d}_a^v \geq \delta$; such a v exists since $\bar{d}_a^{s'} \geq \delta$ and $\bar{d}_a^{u'} = 0$. Let v' be the vertex following the last occurrence of v on P. Note that $\bar{d}_a^{v'} < \delta$, and Constraints (24) inply that $v' \in F$. Moreover, Constraints (23) give $\bar{d}_a^v \leq \bar{d}_a^{v'} + \bar{x}_a^{v'}$. Thus $\bar{d}_a^{v'} < \delta \leq \bar{d}_a^v \leq \bar{d}_a^{v'} + \bar{x}_a^{v'}$.

Now, let y be, as we go on P, the last vertex of F (last occurrence also) such that there exists $b \in B$ with $\bar{d}_b^y < \delta \leq \bar{d}_b^y + \bar{x}_b^y$; such a vertex y exists since v' is a candidate. Note that also $\bar{d}_a^y < \delta$ since otherwise we could go on P from y to u and find v' after y as explained above. Then also $\bar{d}_a^y < 1/(k+1)$ and therefore bank a is loaded in some register at y. As we go on P from y to u, no load instructions are selected by the algorithm after y (as we cannot have vertices $w \in \widehat{P}[y]$ and banks b with $x_b^w = 1$, since this contradicts the choice of y), and thus P is fulfilled.

For the derandomization, we can try polynomially many values of $\delta \in (0, 1/(k+1))$, or prove as in Subsection 3.2 that any such value will do. The following comprises this discussion (as well as that of Section 3):

Theorem 4.1 *There is a $k(k+1)$-approximation algorithm for k-BSIM.*

Theorem 4.2 *There is a polynomial-time algorithm whose output uses at most $2k-1$ registers and a number of load instructions at most $2k$ times the optimum solution with k registers.*

Proof: For this bicriteria result, we choose independently and uniformly at random real numbers $\delta_b \in (0, 1/2)$. Then, for every $v \in F$, if there is an a with $\bar{d}_a^v < \delta_a \leq \bar{d}_a^v + \bar{x}_a^v$, we set $x_a^v = 1$. If $x_a^v = 1$, we load at node v (in some of the available $2k-1$ registers) all the banks b with

$\bar{d}_b^v < \delta_b$; we write this as $\hat{x}_b^v = 1$. Indeed, there can be at most $2k-1$ banks b with $\bar{d}_b^v < 1/2$, from Constraint (29).

We next argue that the necessary trails are fulfilled. Pick an arbitrary $a \in B$, an arbitrary $u \in R_a$, and an arbitrary trail P from s' to u. Let u' be the last vertex of P in F; u' exists since s is on P and in F. From Constraints (25) and (24) we get $\bar{d}_a^{u'} = 0$. From Constraints (21) we get $\bar{d}_a^{s'} \geq 1/2$. Let v be, as we go on P, the last vertex (last occurrence also) such that $\bar{d}_a^v \geq \delta_a$; such a v exists since $\bar{d}_a^{s'} \geq \delta_a$ and $\bar{d}_a^{u'} = 0$. Let v' be the vertex following the last occurrence of v on P. Note that $\bar{d}_a^{v'} < \delta_a$, and Constraints (24) inply that $v' \in F$. Moreover, Constraints (23) give $\bar{d}_a^v \leq \bar{d}_a^{v'} + \bar{x}_a^{v'}$. Thus $\bar{d}_a^{v'} < \delta_a \leq \bar{d}_a^v \leq \bar{d}_a^{v'} + \bar{x}_a^{v'}$.

Now, let y be, as we go on P, the last vertex of F (last occurrence also) such that there exists $b \in B$ with $\bar{d}_b^y < \delta_b \leq \bar{d}_b^y + \bar{x}_b^y$; such a vertex y exists since v' is a candidate. Note that also $\bar{d}_a^y < \delta_a$ since otherwise we could go on P from y to u and find v' after y as explained above. Therefore also bank a is loaded in some register at y. As we go on P from y to u, no load instructions are selected by the algorithm after y (as we cannot have vertices $w \in \widehat{P}[y]$ and banks b with $x_b^w = 1$, since this contradicts the choice of y), and thus P is fulfilled.

Moreover, for any $u \in V$, requiring $a \in B$, any path P from s to u has vertex v' with $\bar{d}_a^{v'} < \delta_a \leq \bar{d}_a^{v'} + \bar{x}_a^{v'}$. Let v be the last vertex on $\widehat{P}$ with $\bar{d}_b^v < \delta_b \leq \bar{d}_b^v + \bar{x}_b^v$, for some $b \in B$. Then also $\bar{d}_a^v < \delta_a$ (as otherwise there is a further, on $\widehat{P}$, vertex v'' with $\bar{d}_a^{v''} < \delta_a \leq \bar{d}_a^{v''} + \bar{x}_a^{v''}$). and therefore bank a is loaded in some register at v. As we go on $\widehat{P}$ from v to u, no further load instructions are selected by the algorithm, and thus at vertex u bank a is loaded.

Let Q_v be the set of banks a with $\bar{d}_a^v < 1/2$. The probability that bank b is loaded at vertex $v \in Q_v$ is (using the independence of the choices of δ_a):

$$Pr[\hat{x}_b^v = 1] \leq Pr[x_b^v = 1] + \sum_{a \in Q_v \setminus \{b\}} Pr[x_a^v = 1] \cdot Pr[\bar{d}_b^v < \delta_b] \leq 2\bar{x}_b^v + \sum_{a \in Q_v \setminus \{b\}} 2\bar{x}_a^v \cdot Pr[\bar{d}_b^v < \delta_b] \quad (30)$$

and thus the expected number of loads at node v is at most

$$\sum_{b \in Q_v} \left(2\bar{x}_b^v + \sum_{a \in Q_v \setminus \{b\}} 2\bar{x}_a^v \cdot Pr[\bar{d}_b^v < \delta_b] \right) = \sum_{b \in Q_v} \bar{x}_b^v \left(2 + 2 \sum_{a \in Q_v \setminus \{b\}} Pr[\bar{d}_a^v < \delta_a] \right)$$

If $|Q_v| \leq k$, then

$$\left(2 + 2 \sum_{a \in Q_v \setminus \{b\}} Pr[\bar{d}_a^v < \delta_a] \right) \leq 2 + 2\left(|Q_v| - 1\right) \leq 2k$$

and thus the expected number of loads at node v is at most $2k \sum_{b \in B} \bar{x}_b^v$.

Otherwise, $|Q_v| \geq k+1$, and Constraint (29) together with $\bar{d}_b^v < 1/2$ gives:

$$\sum_{a \in Q_v \setminus \{b\}} \bar{d}_a^v \geq |Q_v| - k - 1/2. \quad (31)$$

We have

$$\begin{aligned} 2 + 2 \sum_{a \in Q_v \setminus \{b\}} Pr[\bar{d}_a^v < \delta_a] &= 2 + 2 \sum_{a \in Q_v \setminus \{b\}} (1 - 2\bar{d}_a^v) \\ &= 2 + 2(|Q_v| - 1) - 4 \sum_{a \in Q_v \setminus \{b\}} \bar{d}_a^v \\ &\leq 2|Q_v| - 4(|Q_v| - k - 1/2) \\ &= 4k + 2 - 2|Q_v| \\ &\leq 4k + 2 - 2(k+1) \\ &= 2k \end{aligned}$$

where we used Inequality (31) for the first inequality and $|Q_v| \geq k+1$ for the last. Thus in all cases, the expected number of bank loads is at most $2k$ times the LP solution value. □

Assuming $\ln n << k$, the following result is an improvement:

Theorem 4.3 *There is a $O(k \ln n)$ randomized approximation algorithm for k-BSIM.*

Proof: Use the rounding method of the previous theorem, with the interval $(0, 1/(8 \ln n))$ for each δ_a. Let Q_v be the set of banks a with $\bar{d}_a^v < 1/(8 \ln n)$; as above $|Q_v| \leq 2k$. Let Q'_v be the (random) set of banks loaded by the algorithm at v.

Claim 4.1 $Pr[|Q'_v| > k] \leq \frac{1}{n^2}$

Proof: We are setting up a Chernoff bound. We define $d_a = \bar{d}_a^v$, $Q = Q_v$, $q = |Q|$, and $\sigma = \sum_{a \in Q} d_a$. We may assume $q > k$ or else the claim is trivially true. For bank $a \in Q$, define the random variables:

$$Z_a = \begin{cases} 1 & \text{if } d_a > \delta_a \\ 0 & \text{otherwise} \end{cases}$$

Define the random variable $Z = \sum_{a \in Q} Z_a$. Let $p_a = 8 d_a \ln n$ and $p = \frac{\sum_{a \in Q} p_a}{q}$. Let X_a $(a \in Q)$ be the random variables $Z_a - p_a$. Then X_a are mutually independent with $Pr[X_a = 1 - p_a] = p_a$ and $Pr[X_a = -p_a] = 1 - p_a$. Define the random variable $X = \sum_{a \in Q} X_a$. Then X satisfies Assumptions A.1.3 of [4] and therefore Theorem A.1.13 of [4] states that, for any $\alpha > 0$,

$$Pr[X < -\alpha] < e^{-\alpha^2/2pq}. \tag{32}$$

We have that the event $|Q'_v| > k$ is included in the event $Z < q - k$, which is the event $X < (q-k) - \sum_{a \in Q} p_a$. Note that

$$-(q-k) + \sum_{a \in Q} p_a = -(q-k) + 8 \ln n \sum_{a \in Q} d_a \geq 7\sigma \ln n,$$

where we used Constraints (29), which state $\sigma \geq (q-k)$. Note also that $q - k \geq 1$, and thus Chernoff's bound from Equation (32) gives

$$Pr[|Q'_v| > k] < e^{-(7\sigma \ln n)^2/(2 \cdot 8\sigma \ln n)} \leq e^{-2\sigma \ln n} \leq e^{-2 \ln n},$$

which is what the claim requires. □

The expected number of banks loaded at vertex v, computed as in the bicriteria algorithm, does not exceed $(2 + 8k \ln n) \sum_{b \in Q_v} \bar{x}_b^v$, when one uses that Constraint (29) actually gives $|Q_v| \leq k\,(1 + 2/(8 \ln n))$. Thus Markov's inequality gives $Pr[\text{number of banks loaded} > 20k(\ln n) Z^*_{LP2}] \leq 1/2$, where Z^*_{LP2} is the objective value of **LP2**. From Claim 4.1, taken as a union over all v, the probability that there is a vertex loading more than k banks is at most $1/n$. So with probability $1/3$ no vertex is overloaded and less than $20k(\ln n) Z^*_{LP2}$ banks are loaded in total. This concludes the proof of Theorem 4.3. □

4.1 Integrality gap

Let $opt(I)$ be the optimum value for a k-BSIM instance I, $Z^*_{IP2}(I)$ be the optimum value for the constructed **IP2** instance, and $Z^*_{LP2}(I)$ be the optimum value for the **LP2** relaxation. We are unable to find the worst case ratio between $opt(I)$ and either $Z^*_{IP2}(I)$ or $Z^*_{LP2}(I)$. The next theorem relates $Z^*_{IP2}(I)$ to $Z^*_{LP2}(I)$ and is not surprising in view of the connection of k-BSIM to $(k+1)$-uniform Hypergraph Vertex Cover.

Theorem 4.4 *For any $\epsilon > 0$, there exists a k-BSIM instance I with $Z^*_{IP2}(I) > (k+1-\epsilon) Z^*_{LP2}(I)$.*

Proof: We use the first reduction from Section 2, starting with a complete $(k+1)$-uniform hypergraph G with n vertices (and thus $\binom{n}{k+1}$ hyperedges). The number n will be picked large enough (and depend on ϵ). We obtain a k-OBSIM instance, which we call J. In J, we call nodes corresponding to the hyperedges in G *hyperedge-nodes* and call nodes corresponding to vertices in G *hypervertex-nodes*. The hypervertex-nodes come in n groups, where all the $k\binom{n}{k+1}^2$ nodes of I coming from the same vertex of G are in the same group and have the same requirement, and vertices in different groups have different requirements. To simplify some notation, we change J by making the hyperedge-nodes transparent. We further transform J into a k-BSIM instance I using the second reduction of Section 2; this adds for every hypervertex-node v a node v_{in} (it should also add v_{out}, but with no arc leaving v_{out} we ignore it) and then we add s' and construct an **IP2** instance we call I'.

Next we describe a fractional (**LP2**) solution to I'. For every hypervertex-node v, in a group with requirement a (where a is a vertex in G), set $d_a^{v_{in}} = 0$, and for all $b \neq a \in B$, set $d_b^{v_{in}} = 1$. Also set $x_a^{v_{in}} = 1/(k+1)$, and for all $b \neq a \in B$, set $x_b^{v_{in}} = 0$. For every hyperedge-node w, corresponding to a hyperedge e of G, set for all $a \in e$, $d_a^w = 1/(k+1)$ and $x_a^w = k/(k+1)$, and for all $b \notin e$, set $d_b^w = 1$ and $x_b^w = 0$. Set for both s and s' all d-values to be 1 and all x-values to be 0. One can check that this is indeed a feasible **LP2** solution, and its objective is $n \cdot k\binom{n}{k+1}^2 \cdot (1/(k+1)) + \binom{n}{k+1} \cdot (k+1) \cdot (k/(k+1))$. Thus:

$$Z^*_{LP2}(I) \leq \frac{nk}{k+1}\binom{n}{k+1}^2 + k \cdot \binom{n}{k+1} \tag{33}$$

Now assume for a contradiction that

$$Z^*_{IP2}(I) \geq (n-k)k\binom{n}{k+1}^2 \tag{34}$$

does not hold. Let $\bar{d}$ and $\bar{x}$ be this feasible solution. Then there exist $k+1$ groups, from vertices $y_1, y_2, \ldots, y_{k+1}$ of G, such that each has a node v^i, for $i = 1, 2, \ldots, k+1$ such that $\bar{x}_{y_i}^{v^i_{in}} = 0$. G being complete, there is a hyperedge $e = \{y_1, y_2, \ldots, y_{k+1}\}$, and let w be the hyperedge-node corresponding to hyperedge e. For all $i = 1, 2, \ldots, k+1$, Constraints (25) for v^i_{in} and v_i give $\bar{d}_{y_i}^{v^i_{in}} = 0$, and Constraints (23) for w and v^i_{in} give $\bar{d}_{y_i}^{w} = 0$, leading to contradicting Constraint (22) for w, a contradiction to $\bar{d}$ and $\bar{x}$ being a feasible solution.

Thus Equation (34) holds, and together with Equation (33), $n \leftarrow \infty$ and k fixed, we conclude that indeed for any $\epsilon > 0$, there exists a k-BSIM instance I with $Z^*_{IP2}(I) > (k+1-\epsilon)Z^*_{LP2}(I)$. □

5 Conclusion

For k-BSIM we presented approximation algorithms with ratios $k(k+1)$, and $O(k \log n)$. Another algorithm outputs a valid solution with objective at most $2k$ times the optimum for k registers, using $2k$ registers. These results hold in arbitrary input CFGs. Our algorithms can easily be extended to the case when required nodes each has a set of banks $A \subset B$ with $|A| \leq k$, and all banks of A must be loaded.

Our hardness results, that hold for acyclic CFGs as well, are that for any $\epsilon > 0$, an approximation ratio of $k - \epsilon$ is NP-hard, and an approximation ratio of $k + 1 - \epsilon$ is unlikely, as it would imply that Vertex Cover has approximation $2 - \epsilon$.

We leave open the existence of a $O(k)$-approximation. It is not immediate to apply our methods to the following variant of the problem: For every CFG node v that requires bank b, we must select a register $i \in \{1, 2, \ldots, k\}$ and ensure every directed path from s to v loads bank b in register i, and no further loads in register i are allowed. Notice, as for example in Figure 3, that k-OBSIM does not require such an i to be selected.

References

[1] Freescale. `http://www.freescale.com`.

[2] Zilog. `http://www.zilog.com`.

[3] Amit Agarwal, Noga Alon, and Moses S. Charikar. Improved approximation for directed cut problems. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, STOC '07, pages 671–680, New York, NY, USA, 2007. ACM.

[4] Noga Alon and Joel H. Spencer. *The Probabilistic Method (second edition)*. Wiley Interscience, 2000.

[5] Peter Bergner, Peter Dahl, David Engebretsen, and Matthew O'Keefe. Spill code minimization via interference region spilling. In *Proceedings of the ACM SIGPLAN 1997 conference on Programming language design and implementation*, PLDI '97, pages 287–295, New York, NY, USA, 1997. ACM.

[6] D. Bernstein, M. Golumbic, Y. Mansour, R. Pinter, D. Goldin, H. Krawczyk, and I. Nahshon. Spill code minimization techniques for optimizing compliers. In *Proceedings of the ACM*

SIGPLAN 1989 Conference on Programming language design and implementation, PLDI '89, pages 258–263, New York, NY, USA, 1989. ACM.

[7] Preston Briggs, Keith D. Cooper, and Linda Torczon. Coloring register pairs. *ACM Letters on Programming Languages and Systems*, 1:3–13, 1992.

[8] Moses Charikar, Chandra Chekuri, To-yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation Algorithms for Directed Steiner Problems. *Journal of Algorithms*, 33(1):73–91, 1999.

[9] Joseph Cheriyan, Howard J. Karloff, and Yuval Rabani. Approximating directed multicuts. *Combinatorica*, 25(3):251–269, 2005.

[10] Keith Cooper, Anshuman Dasgupta, and Jason Eckhardt. Revisiting Graph Coloring Register Allocation: A Study of the Chaitin-Briggs and Callahan-Koblenz Algorithms. In Eduard Ayguad??, Gerald Baumgartner, J. Ramanujam, and Ponnuswamy Sadayappan, editors, *Languages and Compilers for Parallel Computing*, volume 4339 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin / Heidelberg, 2006.

[11] Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. A new multilayered PCP and the hardness of Hypergraph Vertex Cover. *SIAM Journal on Computing*, 34(5):1129–1146, 2005.

[12] Heiko Falk. Wcet-aware register allocation based on graph coloring. In *Proceedings of the Fourty-Sixth ACM/IEEE Design Automation Conference*, DAC'09.

[13] Changqing Fu and Kent Wilken. A faster optimal register allocator. In *Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture*, MICRO 35, pages 245–256, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.

[14] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Multiway cuts in node weighted graphs. *Journal of Algorithms*, 50(1):49–61, 2004.

[15] Anupam Gupta. Improved results for directed multicut. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, pages 454–455, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[16] Klaus Jansen. The allocation problem in hardware design. *Discrete Applied Mathematics*, 43(1):37–46, 1993.

[17] Klaus Jansen and Lorant Porkolab. On preemptiveresource constrained scheduling: Polynomial-time approximation schemes. *SIAM Journal on Discrete Mathematics*, 20(3):545–563, 2006.

[18] Klaus Jansen and Joachim Reiter. An approximation algorithm for the register allocation problem. *Integration, the VLSI Journal*, 25(2):89–102, 1998.

[19] Sampath Kannan and Todd A. Proebsting. Register allocation in structured programs. *Journal of Algorithms*, 29(2):223–237, 1998.

[20] David Koes and Seth Copen Goldstein. A progressive register allocator for irregular architectures. In *Proceedings of the International Symposium on Code Generation and Optimization*, CGO '05, pages 269–280, Washington, DC, USA, 2005. IEEE Computer Society.

[21] David Ryan Koes and Seth Copen Goldstein. A global progressive register allocator. In *Proceedings of the ACM SIGPLAN 2006 conference on Programming language design and implementation*, PLDI '06, pages 204–215, New York, NY, USA, 2006. ACM.

[22] David Ryan Koes and Seth Copen Goldstein. Register allocation deconstructed. In *Proceedings of the 12th International Workshop on Software and Compilers for Embedded Systems*, SCOPES '09, pages 21–30, New York, NY, USA, 2009. ACM.

[23] Minming Li, Chun Jason Xue, Tiantian Liu, and Yingchao Zhao. Analysis and approximation for bank selection instruction minimization on partitioned memory architecture. *Journal of Combinatorial Optimization*, 23(2):274–291, 2012.

[24] Serge A. Plotkin, David B. Shmoys, and Eva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20:257–301, 1995.

[25] Fernando Magno Quintão Pereira and Jens Palsberg. Register allocation by puzzle solving. In *Proceedings of the 2008 ACM SIGPLAN conference on Programming language design and implementation*, PLDI '08, pages 216–226, New York, NY, USA, 2008. ACM.

[26] Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[27] Michael D. Smith, Norman Ramsey, and Glenn Holloway. A generalized algorithm for graph-coloring register allocation. In *Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation*, PLDI '04, pages 277–288, New York, NY, USA, 2004. ACM.

[28] Mikkel Thorup. All structured programs have small tree-width and good register allocation. *Information and Computation*, 142(2):159–181, 1998.

[29] Javier Zalamea, Josep Llosa, Eduard Ayguad??, and Mateo Valero. Modulo scheduling with integrated register spilling. In Henry Dietz, editor, *Languages and Compilers for Parallel Computing*, volume 2624 of *Lecture Notes in Computer Science*, pages 239–253. Springer Berlin / Heidelberg, 2003.

[30] A. Zelikovsky. A series of approximation algorithms for the acyclic directed Steiner tree problem. *Algorithmica*, 18(1), 1997.

[31] Yuan Zhou. Hardness of register loading, 2010. personal communication.

[32] Leonid Zosin and Samir Khuller. On directed Steiner trees. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'02, pages 59–63, San Francisco, CA, USA, 2002. Society for Industrial and Applied Mathematics.