Color Permutation:
An Iterative Algorithm for Memory Packing

Jianwen Zhu

Electrical and Computer Engineering

University of Toronto

November 6th, 2001

`jzhu@eecg.toronto.edu`

`http://www.eecg.toronto.edu/˜jzhu`

Outline

- *Motivation and problem definition*
- Previous work
- Acyclic orientation
- Color permutation
- Results

Need for Memory Micro-Architecture Exploration

- Opportunity of system-on-chip
  - Integrating memory into logic: SDRAM
  - Integrating logic into memory: IRAM
- Memory size impacts chip performance
  - Area: proportional to #words
  - Speed: proportional to #words per column
  - Power: proportional to #words
- Memory architecture
  - Memory bank partition
  - Memory hierarchy: cache organization

Previous Work on Memory Micro-Architecture Exploration

- Memory layout for cache performance
  - I-cache: Pettis and Hansen (1990), McFarling (1989)
  - D-cache: Lam, Rothberg and Wolf (1991)
  - Embedded software: Panda, Dutt, Nicolau (1998)
- Data memory compression
  - Philip's Phedio project (JVLSISP'95)
  - IMEC's Matisse project (Kluwer'98)
  - UC, Irvine's ACES group (CODES'98)
  - Verbauwhede, Sheers and Rabaey (DAC'94)
  - Zhao and Malik (TVLSI'00)
  - Zhu (DATE'01)
- Rencent Survey: Panda, Cathoor, Dutt et. al. TODES'01 April issue

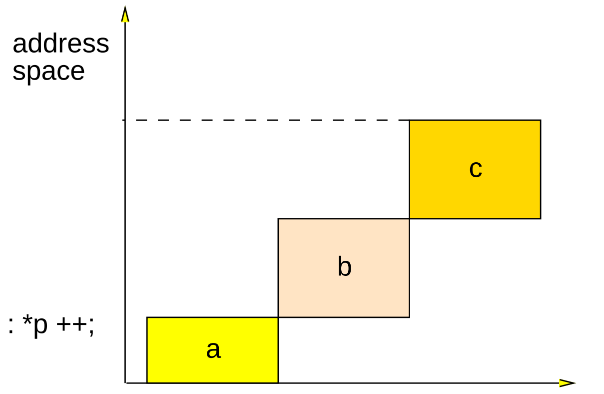
A Motivational Example

```
block a, b, c;

for( i = 0; i < 100; i ++ )
    b[i] = rom[i] * a[i];
for( i = 0; i < 100; i ++ )
    c[i] = b[i] > 255 ? 255 : b[i];
```

(a)

```
block a, b, c;

p = &b;
for( i = 0; i < 100; i ++ )
    *p ++ = rom[i] * a[i];
p = &b; q = &c;
for( i = 0; i < 100; i ++ )
    *q ++ = *p  > 255 ? 255 : *p ++;
```

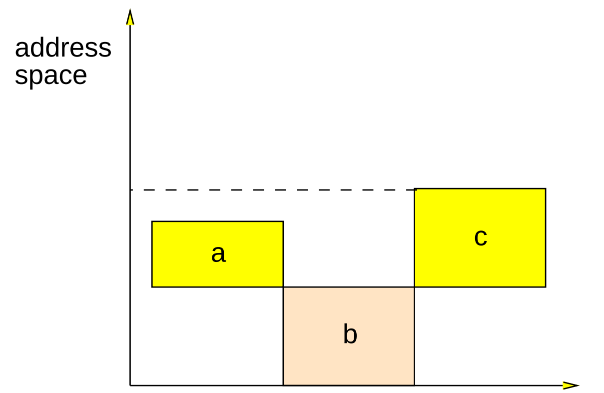
(b)



(c)

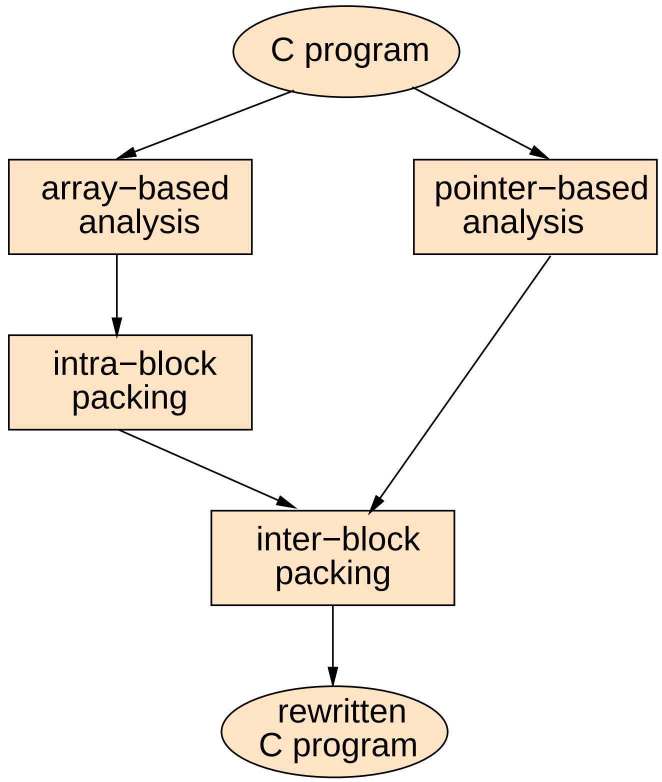
```
union {
    block a, c;
    } cluster1;
union { block b; } cluster2;

for( i = 0; i < 100; i ++ )
    cluster2.b[i] = rom[i] * cluster1.a[i];
for( i = 0; i < 100; i ++ )
    cluster1.c[i] = cluster2.b[i] > 255 ? 255 : cluster2.b[i];
```

(d)



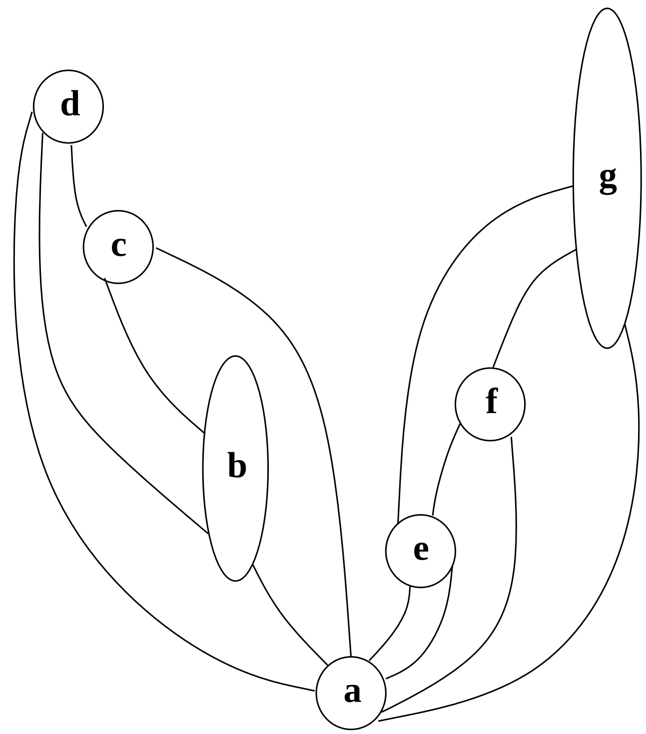
(e)

Memory Allocation Frame Work



Memory Packing Problem

- Given a *Conflict Graph*
  - Nodes V: a set of memory blocks
  - Edges $E \subseteq V \times V$: conflict relationship from analysis
  - Weight $size : V \mapsto Z$: size of blocks
- Find a mapping $A : V \mapsto Z$, such that
  - Correctness: $\langle u,v \rangle \in E \rightarrow [A(u), A(u)+size(u)] \cap [A(v), A(v)+size(v)] = \oslash$
  - Optimality: $max_{v \in V} A(v) + size(v)$ is minimal



Memory Packing by Coloring

- Packing Algorithm

$clr = color(V,E);$

$total = |clr(V)|;$

forall($c \in [0..total-1]$ **)**

$\quad off(c) = max_{clr(v)=c} size(v);$

forall($c \in [1..total-1]$ **)**

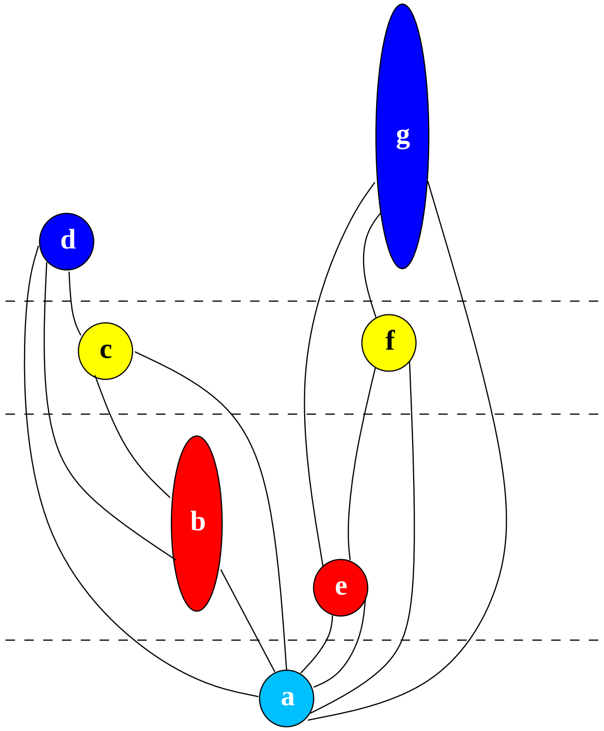
$\quad off(c+1) = off(c) + off(c+1);$

forall($v \in V$ **)**

$\quad a(v) = off(clr(v));$

return a ;

$\}$



	a	b	c	d	e	f	g	total
addr	0	1	3	4	1	3	4	7

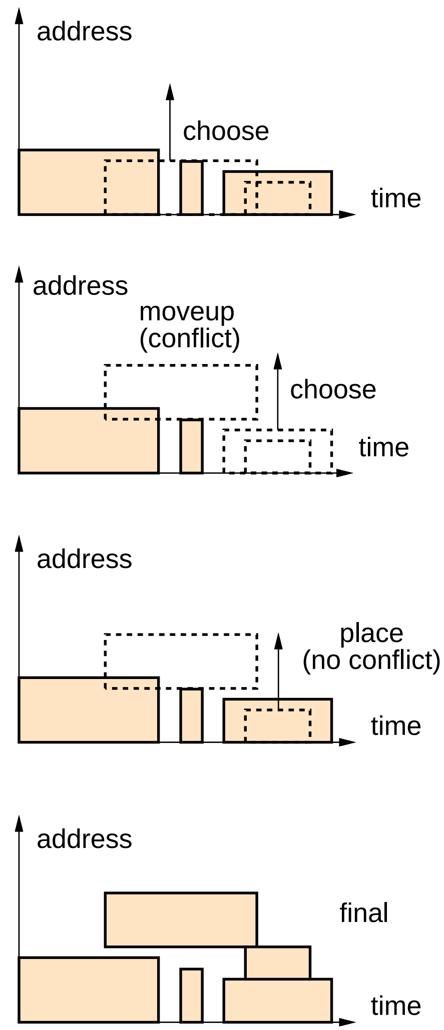
- Pros and Cons
  - Runs in linear time
  - Poor allocation result

Memory Packing by Try and Error

- Packing Algorithm

```
while( todo ≠ ∅ )  {
    v = choose(todo);
    if( check(v) )  {
        todo = todo − {v};
        placed = placed ∪ {v};
        }
    else
        moveup( v );
    }
```

- Pros and Cons
  - Good allocation result
  - Cubical runtime



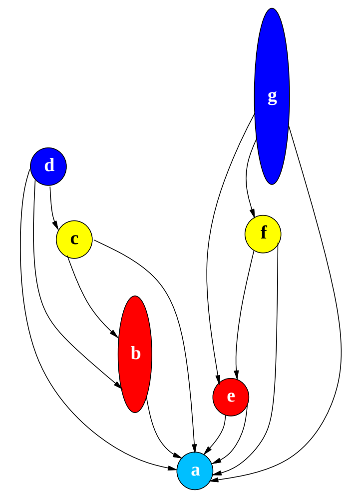
Looking for a Better Approach

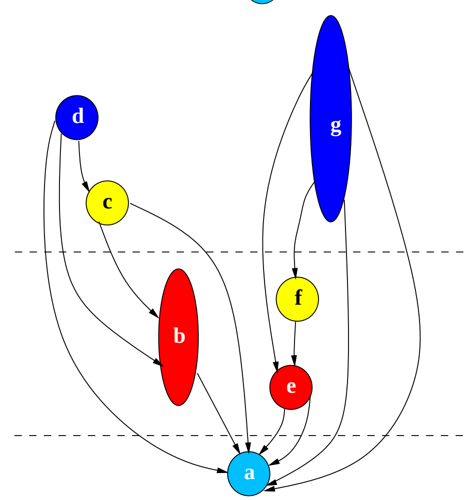
- Problems with try and error
  - Backtracking: each block has to be visited many times
  - Conflict detection
- Desired algorithm
  - Each block is visited only once
  - Conflict constraint implicitly satisfied

Acyclic Orientation

- What if the conflict graph is
  - oriented
  - acyclic
- If address = time, a familiar scheduling problem (ASAP)!
- Theorem: Any valid schedule is a valid allocation
- Theorem: Minimum size is the longest path length of ASAP
- Complexity: ASAP is linear

	a	b	c	d	e	f	g	total
addr	**0**	**1**	**3**	**4**	**1**	**3**	**4**	**6**





Existence of Acyclic Orientation

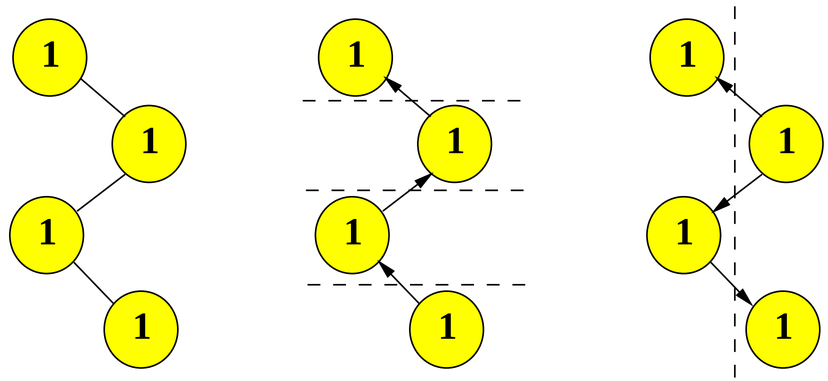
- Does acyclic orientation always exist?

  Theorem: For any permutation $P : V \mapsto Z$ of vertex set V, there exists an acyclic orientation.

- Constructive Proof:

  Let $F = \{\langle u, v \rangle \in E | P(u) < P(v)\}$.

- Does any vertex permutation gives a good allocation?



Iterative Improvement Framework

- Every vertex permutation defines a solution
- The solution can be evaluated in linear time
- Classic strategy to search for good solution
  - Greedy (Hill climbing): $O(h|V| + |E|)$
  - Simulated annealing
  - Simulated evolution

```
bestCost = ∞;
P = [0..|V| - 1];
count = h;
do {
   P = perturb(P);
   F = orient(G,P);
   cost = asap(G,F);
   if( cost < bestCost ) {
      bestCost = cost;
      bestP = P;
      }
   count = count - 1;
   } while( count > 0 ) ;
return P ;
```

P-Admissibility

- Insight into solution space structure
- P-Admissibility (Murato'95)
  - The solution space is finite
  - Every solution is feasible
  - Evaluation is in polynomial time
  - There exists a solution which corresponds to one of the optimal solutions
- Theorem

  For every memory packing $A : V \mapsto Z$, there exists a permutation P from which A can be derived.
- Size of solution space: $n!$, where $n = |V|$

Color Permutation

- Compress the solution space further
- Theorem:

  For any coloring $C : V \mapsto Z$, there exists an acyclic orientation F.

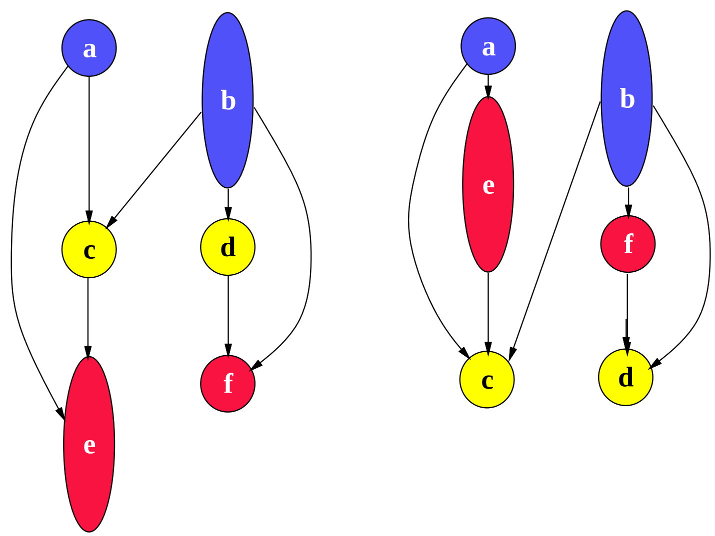
- Constructive proof

  Let $F = \{\langle u, v\rangle \in E \mid C(u) < C(v)\}$

- Solution space: $\chi!$ where χ is the chromatic number of G
- No longer P-admissible

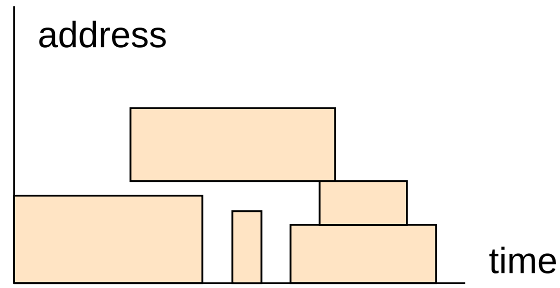
Color Permutation Algorithm and Example

```
bestCost = ∞;
clr = color(G);
count = h;
do {
   clr = perturb(clr);
   F = orient(G,clr);
   cost = asap(G,F);
   if( cost < bestCost ) {
      bestCost = cost;
      bestClr = clr;
      }
   count = count − 1;
   } while( count > 0 ) ;
return bestClr ;
```

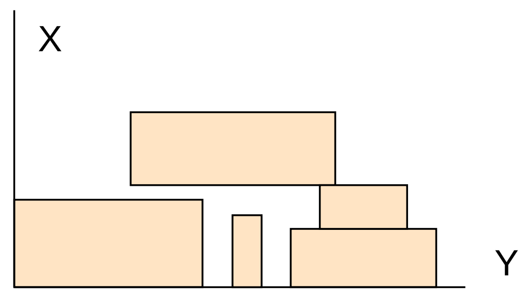


Another Perspective

- Equivalent to *floorplan* problem
  - equate address to X
  - equate time to Y
- Except
  - Constraint on Y dimension (time) is prefixed by application
  - Constraint on Y dimension does not have to be interval graph
- Sequence-pair algorithm



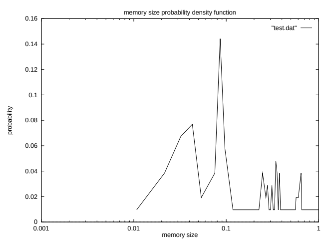
memory packing problem



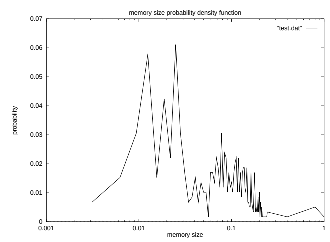
floorplan problem

Benchmarking Methodology

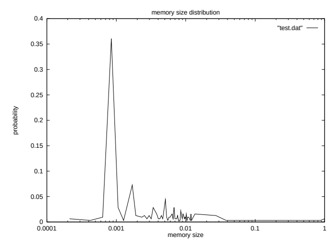
- Problem with benchmarking for previous study
  - No standard benchmark set available
  - Net result for entire framework, no result for packing
- Our approach
  - Standard DIAMCS graph benchmark set
  - Node size randomly generated according to application profiling



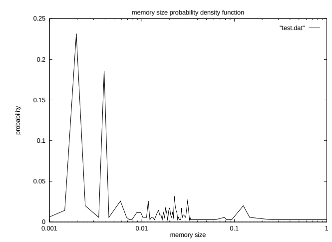
EPIC



JPEG



MPEGDEC



MPEGENC

- Data obtained on Sun Ultra-5.

Experimental Result

Benchmark	# nodes	# edges	total size		runtime (ms)	
			color	perm	color	perm
myciel3	11	20	89792	74688 (16%)	0	0
myciel4	23	71	125120	83520 (33%)	0	10
myciel5	47	236	128064	92736 (27%)	0	40
anna	138	986	297600	156544 (47%)	10	200
david	87	812	296768	201408 (32%)	0	130
le450_15a	450	8168	543296	383296 (29%)	150	14550
le450_5a	450	5714	364160	258112 (29%)	90	3990
le450_5d	450	9757	422080	301568 (28%)	170	16890
queen10_10	100	2940	415744	290560 (30%)	20	1130
queen14_14	196	8372	635456	424640 (33%)	70	13240
queen6_6	36	580	242752	167104 (31%)	0	180
miles500	128	2340	442240	265408 (39%)	20	2230
mulsol	197	3925	940864	686592 (27%)	100	16940
mulsol	184	3916	600384	449216 (25%)	60	5130
inithx	645	13979	761280	481728 (36%)	310	84690
inithx	621	13969	769024	471552 (38%)	290	98280
average improv	31%					

Conclusion

- New memory packing algorithm proposed
  - Acyclic orientation: linear time algorithm for solution evaluation
  - Vertex permutation: a P-admissible solution space of size $n!$
  - Color permutation: a compressed solution space of size $\chi!$
- Experiment shows
  - Good run time: $O(h(|V|+|E|))$
  - Superior result over scalar approach: 30%