Completing High-quality Global Routes

Jin Hu[†], Jarrod A. Roy[‡] and Igor L. Markov[†]

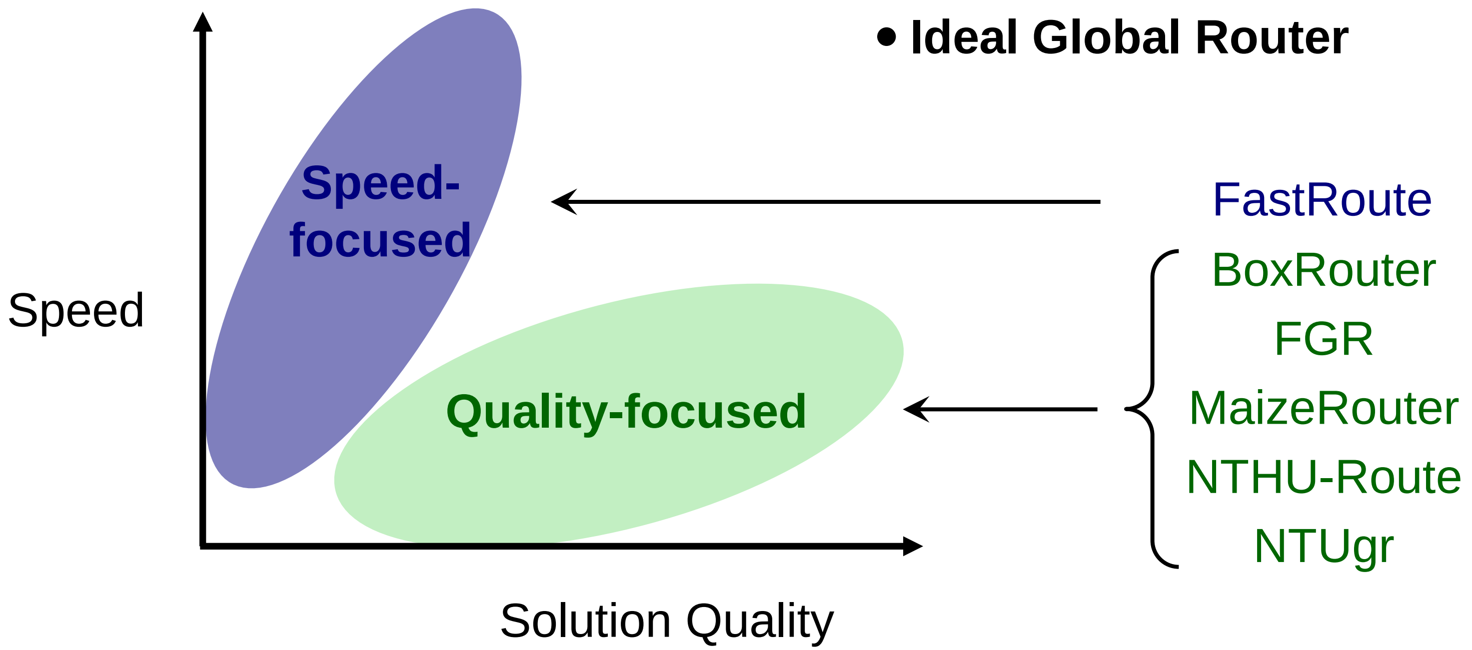
[†]Dept. of Computer Science and Engineering, University of Michigan
[‡]IBM Systems and Technology Group, Austin, TX

Design Flow and Motivation

Global Routing should produce routes:

- That do not cause violations (feasibility)
- That use minimal routing resources (quality)
- In a reasonable amount of time (runtime)

Quality vs. Runtime

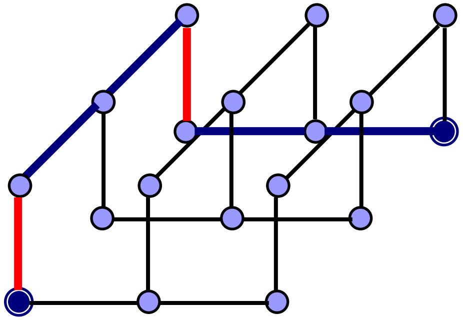


Ideal Global Router

- Robustness, route with **no violations**
- Focus on Solution Quality without sacrificing Runtime

Global Routing Formulation

- Given: Routing Grid and Netlist
- Objective: route all nets while
  minimizing **wirelength** = routed length + number of vias
  subject to **capacity constraints** (no violations):
  - **Violations**: number of nets exceeds edge capacity
  - **Routed length**: total number of segments used on layers
  - **Vias**: number of times route changes layers



Routed length = 4

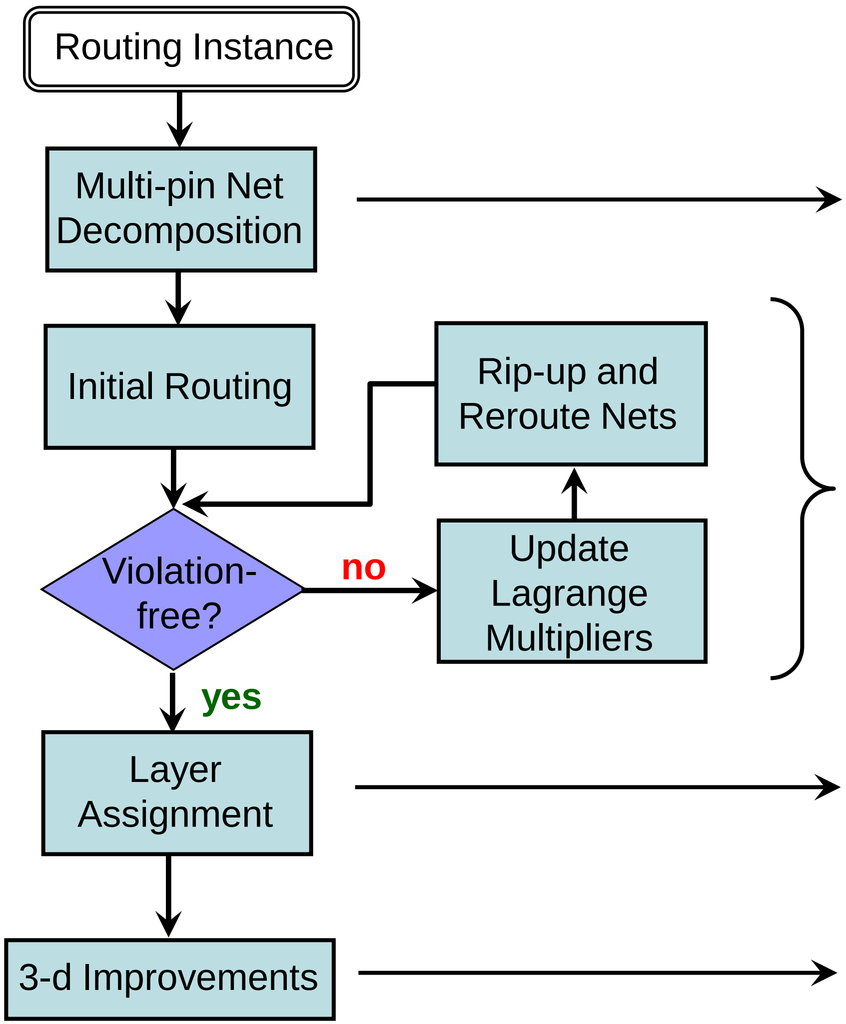
Number of Vias = 2

Total Wirelength = 6

Contributions

- Facilitate robustness:
  Branch-free representation (BFR)
- Techniques for shorter routes
  - Dynamically Adjusting Lagrange Multipliers (DALM)
  - Trigonometric Penalty Function (TPF)
- Techniques to reduce runtime
  - Cyclic net locking (CNL)
  - Aggressive lower-bound estimates for A* (ALBE)

Common Global Routing Flow



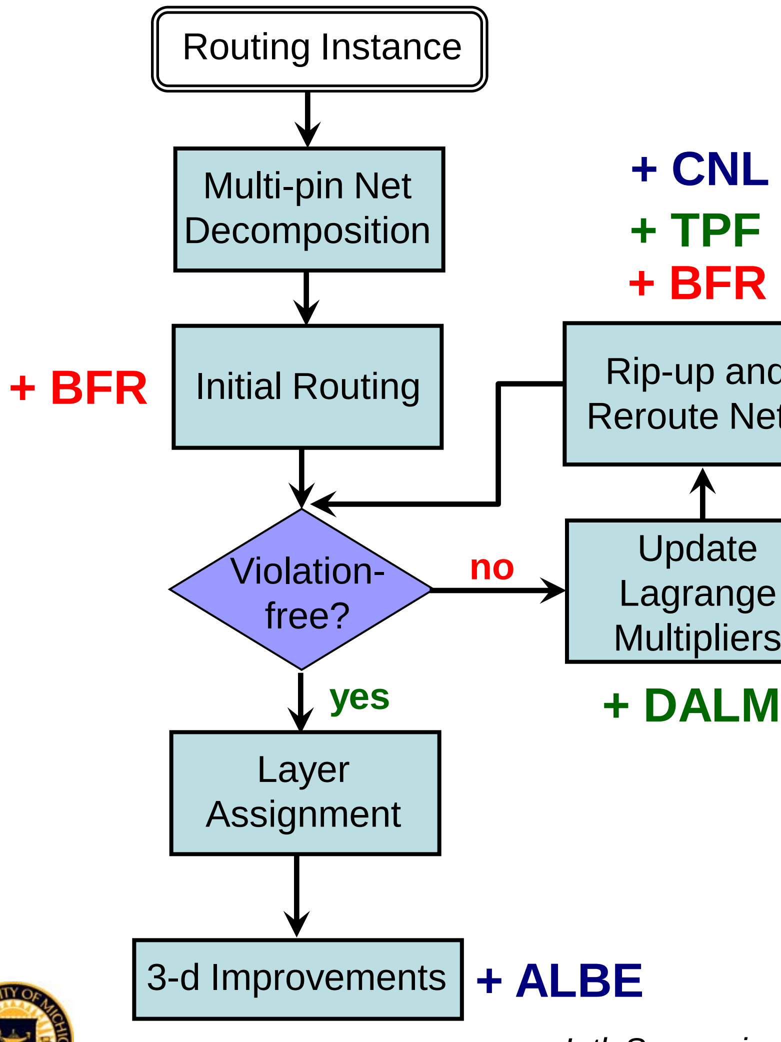
3+ pin nets into 2-pin subnets

1. Route with shortest paths
2. If violations exist, then rip-up nets in violation
3. Reroute nets

2-d routes to 3-d routes

Optional

BFG-R Routing Flow



Robustness

- Branch-free Representation (BFR)

Quality Improvements

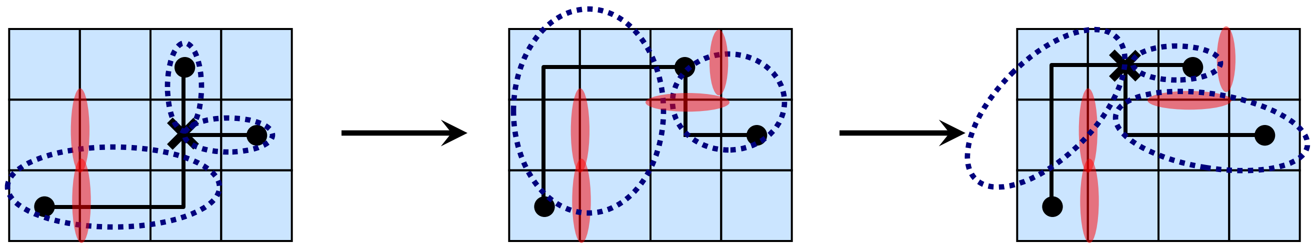
- Dynamically Adjusting Lagrange Multipliers (DALM)
- Trigonometric Penalty Function (TPF)

Runtime Improvements

- Cyclic Net Locking (CNL)
- Aggressive Lower-bound Estimates (ALBE)

Branch-free Representation

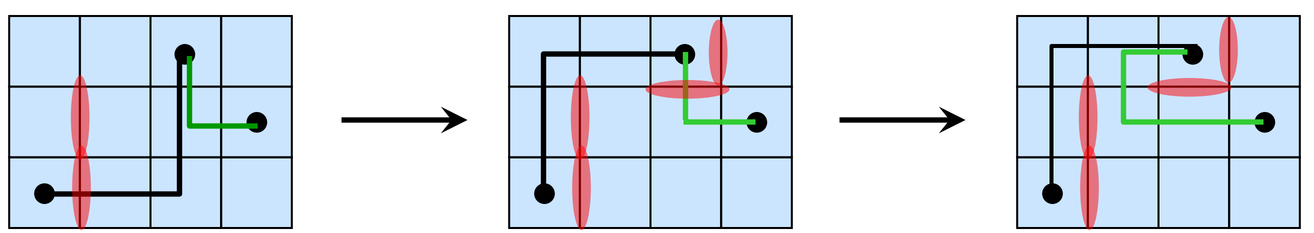
Route of Net *n*



3 segments, 1 branching point | **2 segments, 0 branching points** | **3 segments, 1 branching point**

Local Change → **Global** Effect on data structure

Branch-free Representation: store *edges* of routes in subnets, no Steiner points



2 subnets | **2 subnets** | **2 subnets**

Local Change → **Local** Effect on data structure

Lagrange-based Routing

- Assign every edge e with history-based cost $c_e = b_e + h_e \cdot C_e$, where:
  b_e : base cost of e
  h_e : history cost of e (Lagrange Multiplier)
  C_e : Congestion penalty of e
- <u>Key observation</u>: rates at which h_e and C_e grow affect <u>quality</u> and <u>runtime</u>
  - Faster (larger steps) → ↓runtime, ↓quality
  - Slower (smaller steps) → ↑runtime, ↑quality

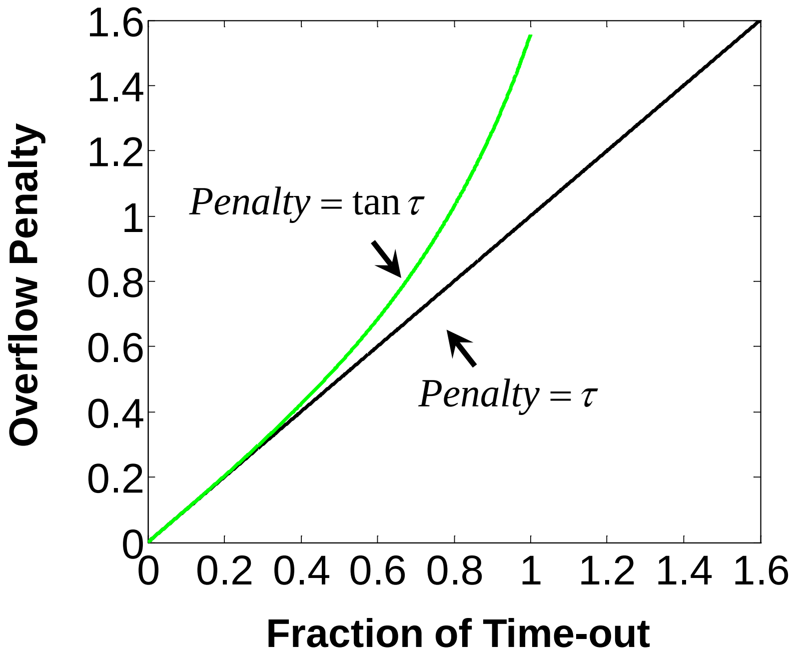
Dynamically Adjusting Lagrange Multipliers

- Key Idea: adjust history cost step based on past violations and wirelength

Previous Iteration	Adjustment to History Cost Increment
↓Violations, ↓WL	None
↓Violations, ↑WL	History cost increment **–=** Δstep
↑Violations	History cost increment **+=** Δstep

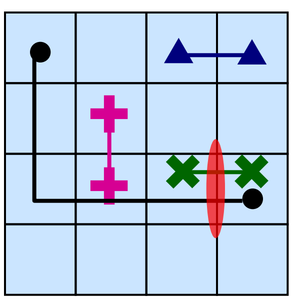
Trigonometric Penalty Function

- Key Idea: encourage ↓ WL early,
  encourage ↓ Violations later

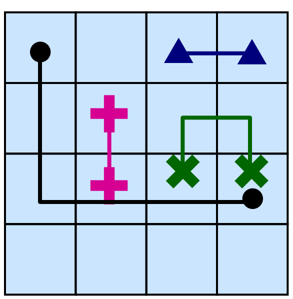


Cyclic Net Locking

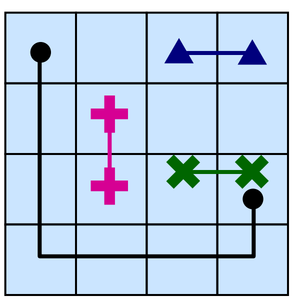
- Key Observation: ↑time spent on larger nets
- Key Idea: route smaller nets more often
- Insight: resolve violations in multiple ways



4 nets, each with minimum length



reroute ✖ first



reroute ● first

Aggressive Lower-bound Estimation for A*-search

- Key Observation:
  after several iterations, $b_e << h_e \cdot C_e$
- Lower bound based on b_e becomes trivial,
  → A* degenerates Dijkstra's algorithm

- Key Idea: Use lower bound based on minimum edge cost of previous route**

**Caveat: No guarantee of shortest path, but does not heavily impact solution quality

Experimental Setup

- Single-core, single thread, 2.8 GHz
- Normalize all routers → same settings for each benchmark
- Changed all benchmark names
  Ex: **`adaptec1`** to **`xXxa_onexXx`**

```
if [$foo==“a1”]
--p2-max-iteration=150
--p2-init-box-size=25
--p2-box-expand-size=1
--overflow-threshold=200
--p3-max-iteration=20
--p3-init-box-size=10
--p3-box-expand-size=15
--monotonic-routing=0
```

NTHU-Route 2.0

```
if( in.IsLevel(3) ||
    in.IsLevel(6) ||
    in.IsLevel(7))
{  Flow1(); }
```

```
if (net_no <= 180000)
{  SetLevel(1); }
else if (net_no <= 200000)
{  SetLevel(2); }
```

NTUgr

```
if ((strstr(benchFile,
   “adaptec1.capo70.3d.35.50.90.gr”)
   != NULL)
{ SLOPE=5; THRESH_M=30; ENLARGE=15;
   ESTEP1=10; ESTEP2=5; ESTEP3=5;
   CSTEP1=5; CSTEP2=5; CSTEP3=10;
   COSHEIGHT=4; VIA=4; A=1;
   L_afterSTOP=1; mazeSet=2;
   goingLV=TRUE; updateType=0; }
```

FastRoute 4.0

Empirical Results – ISPD08

On the 12 known-routable ISPD08 benchmarks

Router Name	NTHU-Route 2.0 (untuned)	NTUgr (untuned)	FastRoute 4.0 (untuned)	Best Tuned	BFG-R (untuned)
Routing Failures	**4**	**2**	**4**	**0**	**0**
WL (0 OF)	**0.99**	**1.04**	**1.01**	**0.99**	**1.00**
Runtime (0 OF)	**1.24**	**4.22**	**0.42**	**0.30**	**1.00**

BFG-R can route all empirically routable benchmarks: 0 routing failures

With high solution quality: <1% difference with best reported

Faster than quality-focused routers NTHU-Route and NTUgr

Empirical Results – *adaptec*

Re-placed *adaptec* designs with `mpl6` with spec'd whitespace %

	NTHU-Route 2.0		NTUgr		FastRoute 4.0		BFG-R	
Benchmark	Cost (e6)	Time (min)	Cost (e6)	Time (min)	Cost (e6)	Time (min)	Cost (e6)	Time (min)
adaptec1, 70%	4.62	7.2	4.83	73.2	**Violations**		4.68	9.8
adaptec2, 60%	5.29	0.9	5.48	3.7	5.31	0.6	5.28	2.2
adaptec3, 80%	**Violations**		**Violations**		**Violations**		12.15	27.2
adaptec4, 80%	10.50	2.3	10.75	9.1	**Violations**		10.49	3.2
adaptec5, 70%	**Violations**		14.44	347.8	**Violations**		13.98	32.6
Average (0 OF)	**1.00**	**0.62**	**1.03**	**5.67**	**1.01**	**0.27**	**1.00**	**1.00**

BFG-R can route all benchmarks: 0 routing failures

Has solution quality ≥ that of other routers

Without sacrificing high runtime

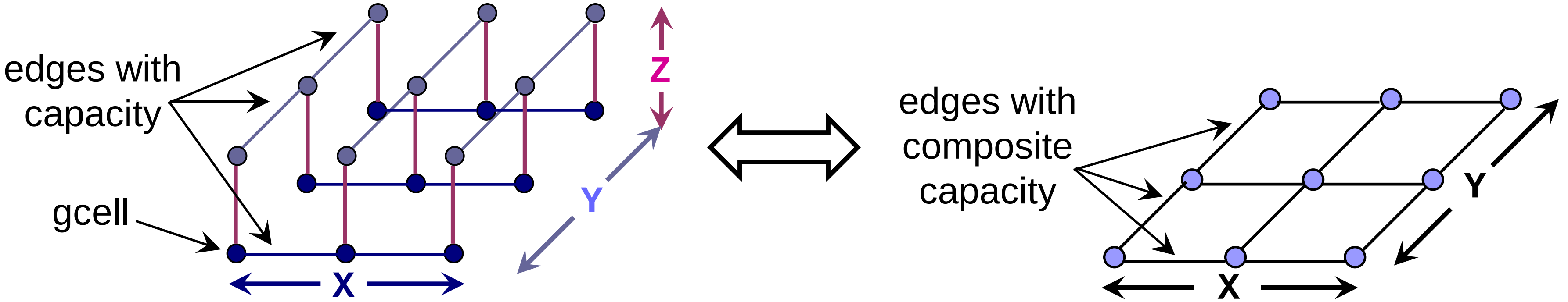
Conclusions

- Presented BFG-R
  - robust software that produces high-quality routes
  - without heavily sacrificing runtime
- Introduced several generic optimizations
  - Facilitates general net topologies
  - Not limited to specific benchmarks

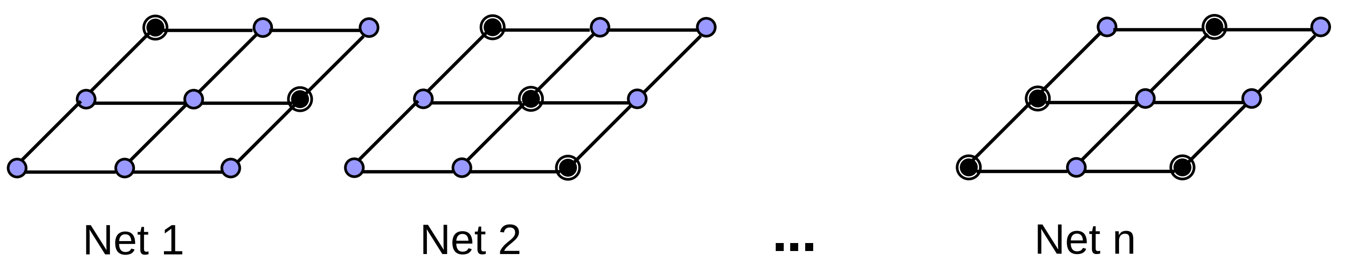
Start Back-up Slides

Global Routing Formulation

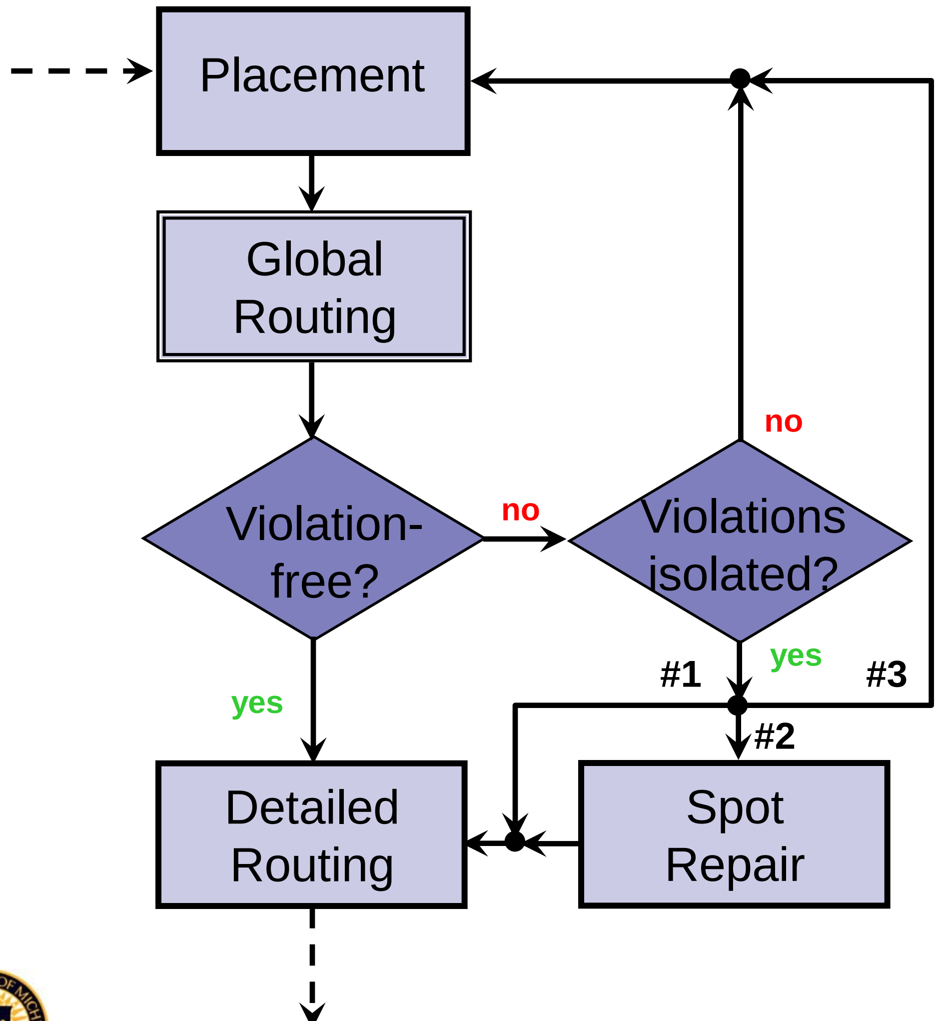
- Routing grid *G*



- Net list *N* with **n** nets



Routing Feasibility



#1: Let detailed router fix violations

#2: Give to secondary tool to fix violations

#3: If too many violations, then must be re-placed

Lagrange Relaxation

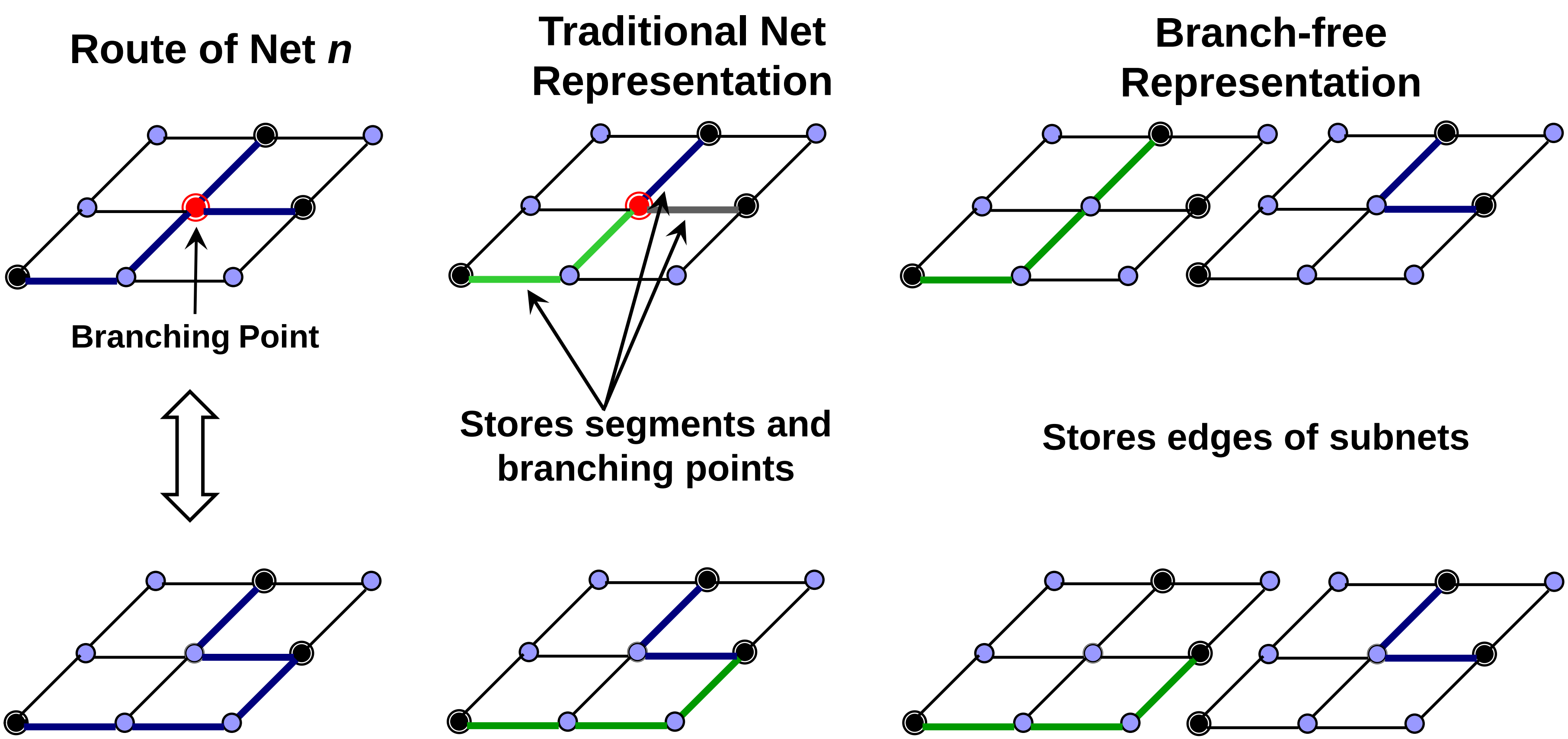
- Optimization problem with constraints:
  minimize total wirelength of nets
  subject to capacity constraints

- Convert constraints to penalties:
  if capacity is exceeded, then edge has increased cost

Lagrange Relaxation

- Add new penalties to objective function
  - Each new penalty has Lagrange multiplier
  - minimize total routed cost of nets
- Optimizing new problem solves original
  - Easier to solve
  - Use iterative methods like rip-up and reroute

Branch-free Representation



Empirical Results – ISPD08

Benchmark	NTHU-Route 2.0 [2]			NTUgr [3]			FastRoute 4.0 [21]			Best Tuned [2, 3, 21]			BFG-R (No Tuning)		
	OF total	Cost (e6)	Time (m)	OF total	Cost (e6)	Time (m)	OF total	Cost (e6)	Time (m)	OF total	Cost (e6)	Router Name	OF total	Cost (e6)	Time (m)
Solution Quality and Runtime for ROUTABLE Benchmarks															
adaptec1	0	5.37	6.4	0	5.67	42.4	0	5.50	3.6	0	5.36	NTHU 2.0	0	5.43	8.4
adaptec2	0	5.24	2.8	0	5.47	7.4	0	5.28	1.2	0	5.23	NTHU 2.0	0	5.23	3.7
adaptec3	0	13.15	4.2	0	13.77	35.0	0	13.26	2.7	0	13.11	NTHU 2.0	0	13.14	16.0
adaptec4	0	12.18	15.1	0	12.41	14.7	0	12.15	1.1	0	12.17	NTHU 2.0	0	12.16	5.2
adaptec5	0	15.54	5.2	0	16.52	100.9	0	15.91	10.3	0	15.54	NTHU 2.0	0	15.67	15.5
bigblue1	0	5.57	10.0	0	5.95	118.3	0	5.89	8.0	0	5.57	NTHU 2.0	0	5.72	10.2
bigblue2	86	9.00	12.2	118	9.47	212.0	Invalid Solution			0	9.06	NTHU 2.0	0	9.11	40.8
bigblue3	32	13.07	9.7	0	13.49	25.6	MAZE RIPUP WRONG			0	13.08	NTHU 2.0	0	13.18	20.6
newblue1	164	4.60	14.2	212	4.82	136.0	542	4.73	13.6	0	4.65	NTHU 2.0	0	4.68	256.9
newblue2	0	7.59	1.1	0	7.85	5.1	0	7.53	0.7	0	7.53	FR 4.0	0	7.57	1.5
newblue5	18	23.14	29.0	0	24.25	117.9	0	23.51	13.8	0	23.17	NTHU 2.0	0	23.30	47.6
newblue6	0	17.70	49.4	0	18.74	76.6	MAZE RIPUP WRONG			0	17.70	NTHU 2.0	0	18.01	15.7
Routing Failures	4			2			4			0			0		
Improv. 0 OF		0.99			1.04			1.01			0.99			1.00	
Solution Quality and Runtime for UNROUTABLE Benchmarks															
bigblue4	256	22.80	72.9	410	24.35	302.9	Invalid Solution			162	23.10	NTHU 2.0	434	23.20	1416.6
newblue3	Time Out			33636	11.00	163.6	38020	10.88	1344.1	31106	17.15	NTUgr	33900	10.64	1420.9
newblue4	222	12.89	31.2	284	13.89	223.3	212	13.16	27.7	138	13.04	NTHU 2.0	218	13.08	1413.3
newblue7	68	35.52	1284.6	906	36.91	1403.9	Invalid Solution			54	35.58	FR 4.0	606	35.21	1421.1

Empirical Results – adaptec

Benchmark	NTHU-Route 2.0 [2] OF total	NTHU-Route 2.0 [2] Cost (e6)	NTHU-Route 2.0 [2] Time (m)	NTUgr [3] OF total	NTUgr [3] Cost (e6)	NTUgr [3] Time (m)	FastRoute 4.0 [21] OF total	FastRoute 4.0 [21] Cost (e6)	FastRoute 4.0 [21] Time (m)	BFG-R OF total	BFG-R Cost (e6)	BFG-R Time (m)
adaptec1, 70%	0	4.62	7.2	0	4.83	73.2	184	5.01	26.4	0	4.68	9.8
adaptec2, 60%	0	5.29	0.9	0	5.48	3.7	0	5.31	0.6	0	5.28	2.2
adaptec3, 80%	38	12.16	19.4	28	12.88	470.0	616	12.74	183.1	0	12.15	27.2
adaptec4, 80%	0	10.50	2.3	0	10.75	9.1	10	10.61	4.8	0	10.49	3.2
adaptec5, 70%	4	13.91	25.2	0	14.44	347.8	628	14.49	50.6	0	13.98	32.6
Routing Failures	2			1			4			0		
Improv. 0 OF		1.00			1.03			1.01			1.00	

Outline

- Methodology
  - Facilitating robustness
  - Improving solution quality
  - Improving runtime
- Experimental Setup
- Empirical results
- Conclusion