RC24509 (W0803-023) March 6, 2008
Computer Science

IBM Research Report

The Coming of Age of (Academic) Global Routing

Michael D. Moffitt
IBM Research Division
Austin Research Laboratory
11501 Burnet Road
Austin, TX 78758

Jarrod A. Roy
The University of Michigan
EECS Department
Ann Arbor, MI 48109

Igor L. Markov
Synplicity Inc.
600 W. California Avenue
Sunnyvale, CA 94086

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

The Coming of Age of (Academic) Global Routing
[Invited Paper]

Michael D. Moffitt[†] Jarrod A. Roy[‡] Igor L. Markov[‡♯]

[†]IBM Austin Research Laboratory, 11501 Burnet Road, Austin, TX 78758
[‡]The University of Michigan, EECS Department, Ann Arbor, MI 48109
[♯]Synplicity Inc., 600 W. California Ave., Sunnyvale, CA 94086
mdmoffitt@us.ibm.com, {royj | imarkov}@umich.edu

ABSTRACT

Wire routing, an important step in modern VLSI design, is increasingly responsible for timing closure and manufacturability. The CAD community has witnessed remarkable improvements in speed and quality of global routing algorithms in response to the inaugural ISPD 2007 Global Routing Contest, where prizes were awarded for best results on a new set of large industry benchmarks.

In this paper, we review the state of the art in global routing and identify several critical techniques that distinguish top routing algorithms. We also discuss open challenges and offer predictions regarding the future of routing research.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids – *placement and routing*
J.6 [**Computer-Aided Engineering**]: Computer-Aided Design
G.4 [**Mathematical Software**]: Algorithm Design and Analysis

General Terms

Algorithms, Design, Performance, Experimentation, Standardization.

Keywords

Benchmarks, Computer-Aided Design, Congestion, Global Routing, Optimization, Routing, VLSI, Wirelength.

1. INTRODUCTION

Despite decades of documented research in global routing [19, 30, 31, 37], commercial routers continue to improve. Today they can handle layouts of unprecedented scale and sophistication. However, the increasing demands for more efficient timing closure and more precise control over manufacturability fuel both academic research and entrepreneurial activity in the industry.

Remarkable progress has been achieved recently in high-performance routing algorithms and software by university researchers. Improving upon simple two-dimensional tools (*Labyrinth* [23] and *Chi* [17]), which laid out a foundation, modern three-dimensional routers – *BoxRouter* 2.0 [10], FGR [36], MaizeRouter [28], *Archer* [32] and *NTHU-Route* [15] – have considerably pushed the envelope with respect to routability, wirelength, and runtime. The Global Routing Contest hosted at ISPD 2007 was largely responsible for encouraging these exciting developments.

To date, the two most authoritative accounts of routing techniques are by Scheffer et al. [37] and by Hu and Sapatnekar [19]; however, the field has progressed considerably since these surveys were written. Moreover, the large industry benchmarks released at the ISPD 2007 contest offer a powerful new means to evaluate and compare ideas in routing, much like the rigorous benchmarking techniques that reshaped the landscape of academic research in VLSI placement several years earlier [1].

Perhaps, least expected by the research community, the emergence of simple yet highly competitive routing engines has cast significant doubt on various sophisticated techniques previously believed to be required for state of the art performance. To this end, we offer an up-to-date survey of leading-edge algorithms, identify those that distinguish best-performing implementations, and attempt to explain why certain techniques fail to produce good results. We also attempt to draw more general conclusions, discuss open challenges, and offer educated guesses about the future of routing research.

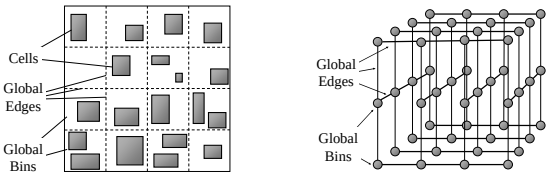
The remainder of this paper is organized as follows. In Section 2, we introduce the global routing problem, along with basic algorithms. In Section 3, we briefly describe progress made in global routing benchmarking. An overview of recent global routers is given in Section 4. In Section 5 we catalog critical components of modern global routers and explain why certain techniques are not worth the effort. In Section 6 we reflect upon lessons learned, and close with predictions for global routing's future in Section 7.

2. BACKGROUND

Global routing is usually formulated as a graph problem and addressed with a suite of graph algorithms.

2.1 Global Routing: Problem Formulation

The problem of global routing can be characterized as follows: there is a grid-graph G specifying a set of vertices V and a set of edges E. As shown in Figure 1, each vertex $v_i \in V$ corresponds to a particular rectangular region (or cell) of the chip, and each edge $e_{ij} \in E$ corresponds to a boundary between adjacent vertices (with a maximum allowable resource m_{ij}). There is also a set of nets N, where each net $n_i \in N$ is composed of a set P_i of pins (with each



(a) Global bin decomposition (b) Corresponding grid graph

Figure 1: The bin decomposition and grid graph of the global routing problem formulation.

pin corresponding to a vertex v_i). A *solution* is a mapping of nets to routes, in which each route connects all the pins of a net using the edges of the graph G.

When evaluating a routing solution (or, for that matter, a routing engine), one is typically concerned with three metrics. *Overflow* refers to the total amount of demand that exceeds capacity over all edges [23]. As it directly corresponds to the routability of the design, overflow is minimized (ideally zero). *Wirelength* is the combined length of segments needed to route all nets, and should also be minimized. In three-dimensional routing, this calculation can also include *vias*, special wires used to connect segments between adjacent layers of metal. Finally, one may be concerned with the *runtime* needed to construct the solution. This is especially true in cases where global routing is repeatedly used to guide a placement algorithm [35]. Global routing is a textbook example of a multi-objective optimization problem, in which the relative importance of the individual criterion depends heavily on the greater context and application.

2.2 Global Routing: Basic Algorithms

Numerous algorithms for global routing are discussed in a comprehensive survey on the topic [19]. However, best-performing routers draw upon only on a subset of these techniques.

Maze routing is a paradigm that seeks the shortest obstacle-avoiding path between two points on a grid with non-negative edge costs. It has long held a reputation as a brute-force approach because it allows all possible paths. Naïve implementations typically employ BFS or Dijkstra's algorithm, whereas A*-search often performs significantly better by guiding the path with (admissible) distance estimates.

Pattern routing restricts point-to-point connections to a small number of fixed shapes, usually minimal-length 'L' and 'Z' paths. Akin to line search, it examines fewer grid edges. However, it provides no local guarantees of optimality, and is often post-processed by maze routing in practice. A less limiting technique is *monotonic routing*, but fits a similar description otherwise.

In addition to point-to-point connections, modern netlists include a number of multi-pin nets which can be routed using Steiner trees. Most academic routers start with near-optimal Steiner trees generated by the stand-alone solver FLUTE [12, 13], which uses look-up tables to produce wirelength-minimal trees for nets containing up to nine pins, and handles larger nets using a divide-and-conquer strategy.

After the initial routes for a set of nets have been determined, one often finds that they overuse some routing resources. Existing routes are then repeatedly torn apart and reassigned in an iterative repair framework known as *Ripup-and-Reroute*. R&R strategies often differ by the order in which to visit nets, as this ordering may significantly impact the allocation of resources.

Routing of multiple nets can be cast as ILP in several ways, of which the most scalable so far has been *BoxRouter*. However, available evidence suggests that ILP-based routers remain far behind R&R routers in speed. Among the more exotic approaches to global routing is its reformulation as a *multi-commodity flow* (MCF) problem [2]. Here, the flow problem is used to solve a linear programming relaxation of global routing, whose dual solution provides a lower bound on the optimum maximum relative congestion.

Due to the computational expense of global routing, some have explored the use of *probability-based congestion prediction* [22, 27, 38, 39] in an effort to help placement algorithms anticipate which regions of the chip will present the most difficult areas for routability. Although recent work has cast doubt on the usefulness of this technique [40], it is still commonly used in industrial placement tools [24].

3. GLOBAL ROUTING BENCHMARKS

▷ **ISPD 1998 Benchmarks [44].** The original standard suite of routing benchmarks – derived from technology that is now a full decade out-of-date – are strictly two-dimensional, and contain a single layer with horizontal and vertical routing tracks. Each edge of the global routing graph has a fixed maximum capacity, expressing a hard physical constraint that any feasible routing must obey.

A refined set of the ISPD 1998 benchmarks [48] were introduced at ICCAD 2007, in which routing capacities have been artificially reduced to increase their difficulty (similar to the technique used in [40] to evaluate methods for probabilistic congestion). Although these modified benchmarks are certainly more difficult than their predecessors (and yet remain manageable enough for reasonable turn-around-times), they also lack obstacles and multiple routing layers, while also including relatively few nets. Future progress in routing is likely to rely heavily on truly up-to-date benchmarks that accurately reflect common instances faced in industrial settings.

▷ **ISPD 2007 Benchmarks [43].** The inaugural Global Routing Contest held at ISPD 2007 introduced a fresh suite of modern global routing benchmarks to the VLSI community. While similar in format to the older set, these test cases reflected a significant leap in difficulty on three different dimensions. First, the sheer size of these instances considerably dwarfed those of the older set, providing an order of magnitude increase in the number of nets and an increase of almost two orders of magnitude in the number of grid cells. Second, the previous assumption that all routing edges share the same global capacity was relaxed with the addition of obstacles. Most importantly, as many as six routing layers were introduced with explicit vias between them, to better match industry practices. High via penalties are indicative of practical routing costs because vias are often thicker than routing tracks and commonly doubled to improve reliability with respect to manufacturing defects. Vias imply additional costs in detailed routing where they render adjacent track segments unusable.

4. MODERN GLOBAL ROUTERS

After the ISPD 2005 Placement Contest, it was feared that a similar contest for routing would not find even one aca-

demic router able to complete an industry benchmark [37]. To the contrary, three academic routers – MAIZEROUTER [28], *BoxRouter* [10], and FGR [36] – proved capable of meeting this challenge, with two more following suit in less than a year [15, 32]. We now review these routers and their precursors.

▷ **Labyrinth [23]** entirely relied on pattern routing. While it does not achieve competitive solutions, it was available in source code and served as a strawman to facilitate subsequent router development and benchmarking.

▷ **Chi [17]** implemented fairly standard R&R and improved it with *congestion amplification*.

▷ **BoxRouter [11] and BoxRouter 2.0 [10].** The basic idea behind *BoxRouter* is to progressively expand a box initiated from the most congested region of the chip, applying an integer linear programming (ILP) formulation to re-route wires between successive boxes. Although the ILP considers only L-shaped patterns for each two-pin decomposition, a round of maze routing is applied thereafter to compute paths for wires that cannot be successfully routed. The improved *BoxRouter* 2.0 uses a dynamic version of A*-search in negotiation-based routing and incorporates *topology-aware wire rip-up* to move wires from congested regions without changing the net topology. The ILP formulation is also extended toward via/blockage-aware layer assignment to handle blockages and guarantee the feasibility.

▷ **FastRoute [33] and FastRoute 2.0 [34].** With respect to runtime, *FastRoute* remains one of the more competitive solvers to date. It uses a congestion map to warp the structure of a Hanan grid [18] during Steiner tree generation, followed by edge shifting and a form of pattern routing. It has also recently been enhanced with monotonic routing and multi-source multi-destination maze routing [34], although the version of *FastRoute* competing at ISPD 2007 did not place highly.

▷ **DpRouter [5].** *DpRouter* is based upon a congestion-aware algorithm that combines two principal techniques: a dynamic pattern routing method to achieve optimal routing solutions for two-pin nets, and a *segment-move* technique to extend its search space (not dissimilar to the edge shifting performed by *FastRoute*).

▷ **Archer [32].** The *Archer* router employs a spectrum of point-to-point routing techniques, ranging from relatively cheap operations (e.g., pattern routing) to expensive but flexible procedures (e.g., traditional maze routing). For a given 2-pin connection, the specific technique used depends on congestion histories. Steiner trees are modified dynamically using a novel Lagrangian formulation for topology optimization. While *Archer* exhibits competitive runtime and routability, it significantly lags behind best-known results for three-dimensional instances and trails by a few percent in the two-dimensional category.

▷ **FGR [36]** — a "Fairly Good Router" — extends the PathFinder router originally developed for FPGAs [26] to handle the scale and sophistication of an ASIC environment with multiple routing layers. It offers several technical novelties, such as a particular function for congestion penalty, and closely linked algorithmic innovations, such as ε-sharing in conjunction with continual net restructuring, and fast layer assignment followed by a 3D clean-up. FGR won the 2D category of the ISPD 2007 Global Routing Contest.

▷ **MaizeRouter [28].** Two elementary edge-based operations – *extreme edge shifting* and *edge retraction* – form the high-level operations of MAIZEROUTER. These techniques are supported by an underlying foundation of *interdependent net decomposition*, in which routing solutions are implicitly maintained by flat collections of intervals (instead of explicitly-defined topologies). Rather than operate on entire nets, individual segments are manipulated one-at-a-time, enabling support for cheap incremental operations. MAIZEROUTER won the 3D category of the ISPD 2007 Global Routing Contest.

▷ **NTHU-Route [15].** Based on iterative rip-up and reroute, *NTHU-Route* uses a history-based cost function to distribute overflow, employing a congested region identification method to specify the order in which rip-ups are performed. Wirelength reduction is achieved through an adaptive multi-source multi-sink maze routing method, whereas overflow reduction is obtained through a refinement process after an optimization bottleneck has been reached.

	Labyrinth [23]	*Chi* [17]	*BoxRouter* [11]	Müller [29]	*FastRoute* [33]	*FastRoute* 2.0 [34]	*DpRouter* [5]	*Archer* [32]	*BoxRouter* 2.0 [10]	FGR [36]	MAIZEROUTER [28]	*NTHU-Route* [15]
Pattern routing	✓		✓		✓	✓	✓	✓	✓		✓	✓
Monotonic routing						✓	✓	✓				✓
Maze routing	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
A*-search								✓	✓	✓	✓	
FLUTE dependence			✓		✓	✓	✓	✓	✓		✓	✓
Topo reconstruct.					✓	✓	✓	✓		✓	✓	✓
Incrementality								✓	✓	✓	✓	
Edge "sliding"						✓			✓		✓	
Resource sharing					✓	✓				✓	✓	
ILP or MCF			✓	✓					✓			
Congestion manip.		✓			✓	✓	✓	✓		✓	✓	✓
History-based								✓	✓	✓	✓	✓
Layer assignment				✓				✓	✓	✓	✓	✓
Open source	✓								✓	✓	✓	

Table 1: Techniques used in several academic global routers published since 2002.

5. ANATOMY OF A GLOBAL ROUTER

If one takes a step back from the nuances of the aforementioned global routers, many of the techniques employed fit into distinctive bins. Table 1 identifies some of the broader characteristics in all major routers to date, separating point-to-point algorithms from other considerations. While some methods are variations on well-known ideas and principles, a few particular similarities suggest the beginning of "new conventional wisdom" in routing technology. Many of the sophisticated techniques found in previous literature [19] are absent from this new conventional wisdom, indicating that theoretically attractive concepts do not necessarily lead to competitive implementations.

In this section we summarize some of the algorithmic themes shared by many successful routers.

5.1 Single-net Tree-topology Generation

A large fraction of academic research in routing focuses

on single-tree routing for wirelength optimization. For instance, *FLUTE* [12] computes optimal Steiner trees for up to 9 pins, and GeoSteiner [42] for any number of pins (but much slower). Indeed, all published routers built since 2006, except FGR, rely upon *FLUTE*.

However, even when considering individual nets, solutions produced by stand-alone constructors cannot be considered truly optimal in the presence of vias and non-isotropic layers. For instance, all pins in the ISPD 2007 benchmarks lie on the bottommost horizontal layer, and optimal solutions are thus typically biased toward trees that avoid purely vertical pin-to-pin connections. Obstacles present yet another complication, although recent work has made significant advances to handle such blockages [25].

Furthermore, the majority of published Steiner-tree algorithms operate on uniform-cost grids, and restrict their search to the Hanan grid according to the Hanan theorem [18]. In contrast, congestion-driven routers operate on a routing grid with non-uniform costs (either from historical factors, a.k.a. Lagrange multipliers, or from congestion amplification), which make the Hanan theorem inapplicable. In practice, costs in neighboring gcells can differ by several times and more, meaning that the Hanan theorem is not true even in the approximate sense. Available heuristics for the non-uniform cost Steiner-tree problem appear much slower and can only be invoked once in many routing iterations [32].

A simple alternative to Steiner trees – MSTs – suffices to achieve best-known results for modern benchmarks [36]. This may seem surprising given that MSTs typically do not offer good approximations to SMTs, however, two factors make MSTs competitive. First, accounting for congestion when building an MST does not require a change in algorithms because existing algorithms can work with edge weights (Prim's algorithm is typically used with several optimizations). Second, and most importantly, continual net restructuring during rip-up and re-route (or other meta-heuristics) appears extremely effective in practice as it closely tracks congestion. Even for routing instances that are fairly uncongested, the initial routing with MSTs produces congested gcells, triggering the process mentioned above, which, if carefully implemented, often results in routes that are very close to optimal.

At the same time, Steiner trees constructors can also be useful in a modern flow. Among several open-source Steiner-tree constructors, *FLUTE* remains the most popular. Its main advantage seems to be its phenomenal speed, facilitated by a large look-up table that takes several minutes to precompute. Additionally, our experience indicates that in many designs the majority of nets do not need to circumvent obstacles during global routing, and a large fraction of nets do not detour.

Nevertheless, the increasingly common use of continual net restructuring during global routing iterations may decrease the value of future research on stand-alone Steiner-tree construction. New research must be validated by direct improvements to final results of a complete global router rather than by statistics on a collection of pointsets, as is currently common in the literature.

5.2 Resource Sharing Among Multiple Nets

Recall that pin-to-pin routing algorithms (pattern routing, A*-search, etc.) assume that multi-pin nets are decomposed into direct connections in some manner. We note that such individual subnets are not truly independent, as they can share net segments to decrease costs. Both FGR and MAIZEROUTER factor in the presence of existing resource requirements of a net when routing its sub-components, thus taking into account the interdependence of resource usage. In FGR, this is done by way of ε-based A*-search, while in MAIZEROUTER, conditional checks for subnet existence are performed in both maze routing and extreme edge shifting. *FastRoute* performed a limited form of edge shifting, but imposed subnet existence as a hard *constraint* rather than a modification of *cost* (i.e., the presence of routing segments on either side of a newly formed pin-to-pin connection is a strict prerequisite). In our experience, this resource sharing is critical – without it, the cost of a shortest path can be severely misrepresented. Arguably, any algorithm that does not incorporate this technique (including the bulk of previous work) is not truly computing shortest paths at all.

5.3 History-based Cost Functions

A clear trend in the field is the adoption of history- or negotiation-based cost functions by global routers. Penalty for using a particular routing edge is gradually increased over time as a function of its demand; specific formulas differ among routers. MAIZEROUTER is an exception to this trend, but it does increase a *global* cost for capacity violations, and thus achieves a similar effect (with less precision).

One might wonder why it has taken more than a decade for negotiation-based algorithms to become popular in academia. Even simple improvements to shortest-path algorithms, such as A*-search, seem to be lacking in previous tools. To this end, we point out that efficient implementations of negotiation-based routing and A*-search require a priority queue, which academic researchers and graduate students may not be willing or able to implement efficiently. However, a highly optimized, reusable, heap-based implementation is now available with all C++ compilers as a part of the Standard Template Library (STL). Moreover, the interface to such priority queues is an international standard. The use of these templates may significantly decrease develoment time and obviate homebrew classes, such as that within the source code of *BoxRouter* 2.0 which duplicates the functionality of the generic `priority_queue` template.

5.4 Incremental Tree Restructuring

Fast incremental restructuring of Steiner-tree topologies appears critical for high-performance routers that route one net at a time, such as FGR and *Archer*. Even MAIZEROUTER, which exclusively routes individual segments, and *BoxRouter* 2.0, which performs a limited form of local wire rip-up, depend heavily upon incremental computation. Clearly, rebuilding nets from scratch is expensive [32]. Furthermore, techniques that instead recompute entire trees are likely to change gcell occupancy considerably, possibly causing unnecessary modification of routed segments that would otherwise remain untouched.

5.5 Internal Models and Representations

To accommodate the incremental restructuring described in the previous section, a global router must employ a reasonably flexible model of its solution. While the topic of internal routing representations has been largely neglected in literature to date, specifics have begun to emerge on how modern routers achieve dynamic model maintenance.

For instance, the breakdown of nets using *interdependent net decomposition* in MAIZEROUTER [28] is one way to ensure that the addition and removal of segments can be cheaply performed regardless of their impact on topology. Aside from enabling topological reconstruction, such representations offer relatively simple-to-understand methods for manipulation and maintenance, and are thus much easier to modify and optimize during development.

5.6 Resource Recovery

The competing objectives of wirelength and overflow indicate that global routing may be difficult to solve using traditional single-objective optimization. However, routability can be viewed as a hard constraint when zero overflow is clearly achievable. In these cases the problem falls into a more familiar mono-criterion paradigm, seemingly eliminating any need to maintain a balanced tradeoff. Many global routers have exploited this idea, e.g., *BoxRouter*'s Post-Routing phase and the last phase of FGR's flow. Similarly, MAIZEROUTER continually performs edge retraction in intermediate phases to collapse extraordinarily long detours. Generalizing from these best practices, we distinguish *resource recovery* mechanisms – techniques that strictly preserve overflow or routability while "reeling in" wirelength. Such recovery may prove useful even when complete routability has not yet been achieved, especially if the retraction of long wires serves to reduce the demand and potentially improve overflow. However, how often to inject these into the flow and at what level of effort remains unclear in general and requires careful experimentation in practice.

5.7 Layer Assignment

Starting in 2007, published routers have consistently begun to incorporate some form of layer assignment into their overall flow. In this regard, the academic community has arrived somewhat late to the party, as the practice of global routing without a layer assignment step was deemed to a *degenerate* flow in 2005 [40]. Nevertheless, the assignment of nets (and, more specifically, of individual routing segments) to layers is a necessary component of any legitimate router.

All recent routers employ a form of *layer projection*, in which segments are allocated to layers only after the feasible solution to a 2D variant of the problem is computed. FGR also supports full-3D routing and a full-3D clean-up phase, but at a considerable runtime penalty. *BoxRouter* 2.0 implements a sophisticated ILP technique to restore 3D routes from 2D projections. It is also possible to optimize 2D routing for eventual 3D restoration. For example, the maze routing phase of FGR does separate segments into horizontal and vertical components, and can therefore easily model the vias connecting segments between this pair of layers. The added benefit of this approach is that vias add to wirelength (and, as shown in [36], their contribution is significant). As a result, there is an important distinction between *via-oblivious* and *via-aware* routing [41], even if the final allocation to specific metal layers is postponed.

Recent strategies for layer assignment, except for full 3D routing in FGR, do not yet account for different routing pitches and wire sizes on metal layers. Since such layer-dependent characteristics are becoming exceedingly common in designs 65nm and smaller [3], this represents a significant disconnect from true physical models, and thus an attractive research challenge.

Global Router	Lines of Code	*FLUTE*	ILP
BoxRouter 2.0 [48]	12,986	✓	✓
Labyrinth [44]	6,556		
FGR 1.0 [49]	3,621		
MAIZEROUTER [50]	2,048	✓	

Table 2: Lines of code in recently released open-source routers and Labyrinth (not including comments), along with dependencies on external packages for Steiner-tree construction (*FLUTE*) and ILP-solving. External packages are not included in line counts. The Flute2.5 distribution includes over 3,000 lines of source code, much of which implements utilities and I/O. The router packages also include a significant fraction of I/O code.

6. LESSONS LEARNED

Several trends can be observed from the dynamics of research in routing and from available empirical results.

6.1 Research in Routing vs. Placement

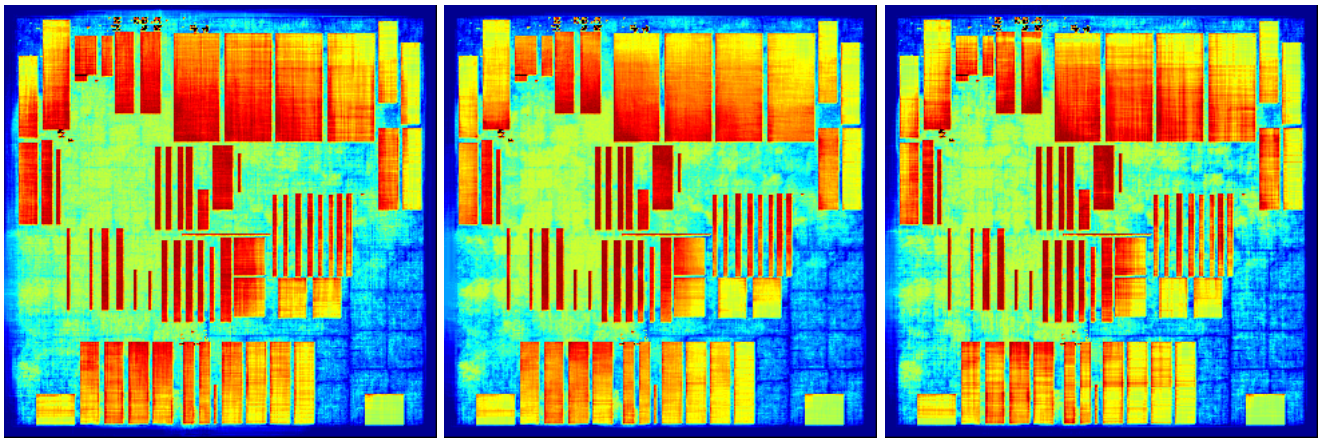
The effort involved in building a successful router appears to be markedly different from that of building a placer. According to the literature and contest results, there seem to be relatively fewer successful algorithmic frameworks for global routing. Furthermore, the amount of code needed to implement a competitive global router is relatively small (i.e., the winning entries required only a few thousand lines of code each), in sharp contrast with current trends in global placement.

However, progress in routing has repeated earlier developments in placement where the literature published before a concerted benchmarking effort was inconclusive as to which algorithms perform better [1]. A common focus on a modern benchmark suite with a clear objective ensures more trustworthy evaluation and encourages algorithm development. For instance, consider that prior to the Global Routing Contest, no routers could successfully route all ten ISPD '98 benchmarks, whereas routers published thereafter [10, 15, 28, 32, 36] solve the entire suite almost effortlessly.

6.2 Simplicity is Key

The Occam's Razor principle stipulates that all things being equal, the simplest solution is often the best. This approach has not been entirely embraced in physical design. The multitude of competing optimization problems faced in design automation have led to complex industry tools assembled into hard-to-control design flows. On the other hand, academic publications often suffer from a bias toward unnecessary obfuscation and formalization, perhaps, encouraged by reviewers. However, for the problem of global routing, the premise that simple algorithms work better than complicated ones appears to hold particularly true. As reported in [16], the top entries in the 2D and 3D categories of the competition were designed from scratch in one month or less, and outperformed engines that had been in development for ten months or more.

Some recent techniques provide interesting guarantees, yet may be ineffective because they impose restrictions on the freedom of the router. For instance, monotonic routing, based on dynamic programming, has gained popularity in some routing tools [5,34], but has not been adopted exclusive-



(a) MaizeRouter (b) FGR (c) *BoxRouter*

Figure 2: Congestion maps for contest solutions to `adaptec1`.

ly by any of the leading modern routers.[1] In contrast, the well-known boxed A*-search [37] that accounts for via costs typically runs faster because it explores fewer grid squares, and is guaranteed to find solutions that are as good or better [36], especially when accounting for via costs.

The lines of code (LOC) required by an application is a fairly common yardstick for measuring its overall complexity (except for cases when it can be intentionally skewed). Table 2 compares the LOCs in *Labyrinth*, *BoxRouter* 2.0, MaizeRouter, and FGR— the routers available in source code. *BoxRouter* is the largest, and also relies upon two significant external applications (*Flute* and an ILP solver). *Labyrinth* is almost half this size, but is significantly out-of-date and does not represent the state of the art. FGR is completely standalone, and requires under four thousand lines of code (almost half the size of *Labyrinth*), of which a quarter are used for I/O. MaizeRouter is the smallest with just over two thousand lines, although it does depend on *Flute*. Since source size is inversely proportional to maintainability, smaller applications are typically preferred for long-term continual research.

We conjecture that MaizeRouter and FGR are so compact due to (1) their reliance on simple yet powerful and robust algorithms, that do not need patching and additional stabilization, (2) their extensive use of the C++ Standard Template Library. Templates for `priority_queue` (heap) and `vector` (expandable array) are heavily used by these routers, whereas *BoxRouter* 2.0 largely implements its own utilities and infrastructure. Anecdotal evidence suggests that the latter is also the case with programming infrastructure at established EDA companies. Such infrastructure was typically developed before common implementations of C++ STL have matured and were extensively optimized for performance, as well as ease of use.

The above consideration about programming infrastructure can be valuable for start-up companies that must quickly build robust infrastructure. In this context, judicious use of standard libraries helps avoiding time-consuming debugging and makes the code more standard, high-level, and readable.

[1] *Archer* [32] does consider monotonic routing as one of several routing templates, though its effectiveness as an individual technique has not been confirmed.

6.3 Ingredients vs. Recipes

Individual techniques – including monotonic routing, pattern routing, edge shifting, etc. – form a veritable toolbox of algorithmic elements. As modern routers continue to mature, it has become clear that the interaction and integration of these individual *ingredients* can play a major role in the success of a complete routing *recipe*.

Whether the construction of such recipes constitutes research, or development, is a matter of debate. However, as routers continue to push toward open-source platforms, it will become easier to plug-and-play with different architectural models (e.g., internal solution representations) with combinations of design decisions (e.g., choices in congestion functions, routing algorithms, and termination criteria). A clean separation of these concepts is not easy, but will allow for faster development of prototypes, and enable an open standard for routing technology.

Note that the choice of low-level techniques and high-level recipes can impact more than scalar metrics of quality such as overflow and wirelength; see Figures 2, 3, and Appendix A for a comparison of routing solutions.

7. CONCLUSIONS AND PREDICTIONS

Extrapolating from current trends, we offer the following predictions for the next five years.

▷ **Benchmarks.** New, larger, open benchmarks will become available, including those for detailed routing, and those with 45nm design rules.[2] As many as nine metal layers were present in modern designs three years ago [40], and it is expected that this number will continue to increase. Hence, global routers may have to tackle much more complex and difficult problems.

▷ **Algorithms.** As routers continue to mature, it is likely that even the most sophisticated projection-based layer assignment schemes may be dominated in quality by a fully 3D aware flow (or, at the very least, a flow that more strongly couples 2D and 3D approaches). This will become true especially as benchmarks begin to express different wire thicknesses, pitches, and propagation delays on layers, as well as

[2] The first company to release such benchmarks would certainly establish a recognizable name in the Physical Design research community.

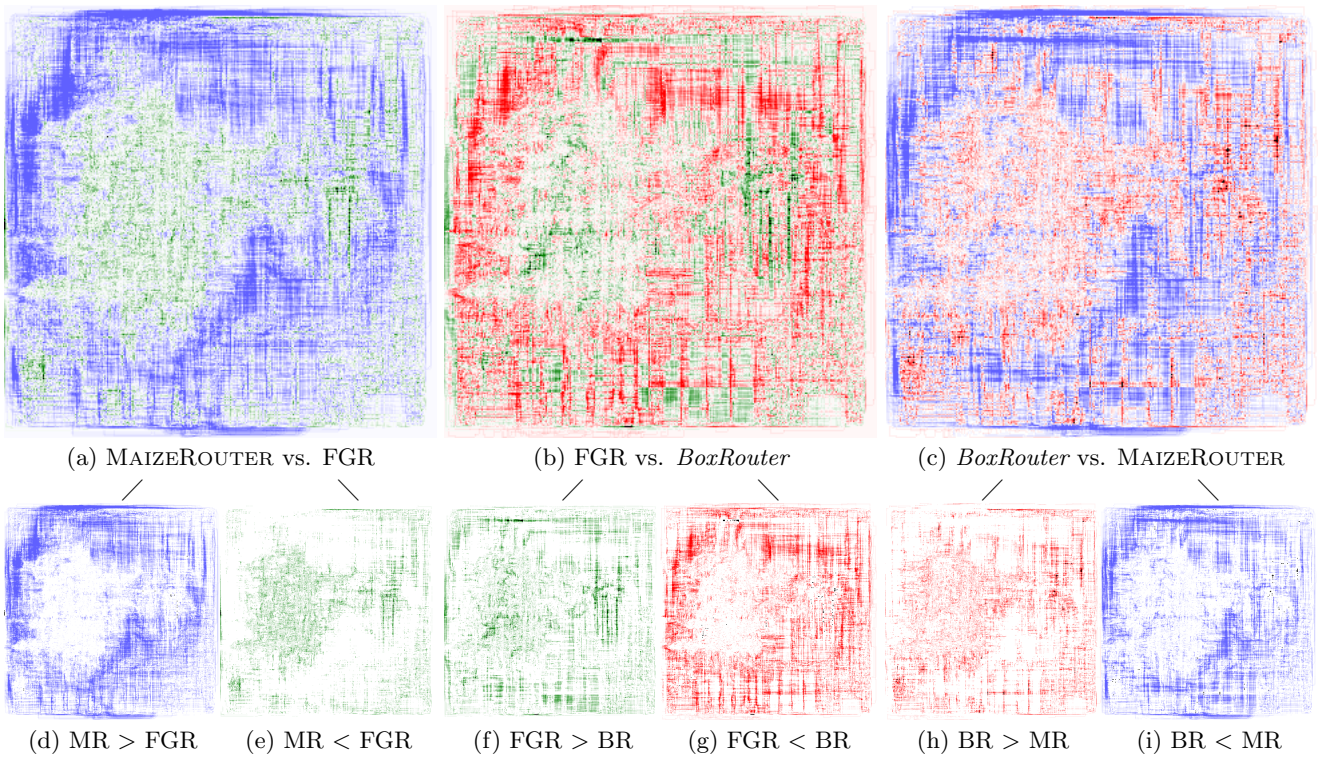


Figure 3: (a-c) Difference maps between routers. (d-i) One-sided difference maps for readers of black-and-white copies.

a wider range of via costs. In such a context, ILP formulations for layer assignment (such as that used by *BoxRouter*) can be more useful than presently, but alone will probably not be sufficient. Parallelization will almost surely become common, with greater success than it has yet to achieve in placement. As hypothesized in [3], more emphasis will be given to timing-driven global routing algorithms.

Multi-level techniques, now commonly used in placement [4,7,21], have not proven as effective in global routing as flat techniques. However, their counterparts in gridless and detailed routing [8,9,14] will likely continue to offer significant runtime savings.

▷ **Open-Source Tools.** Building on the newly released open-source global routers, fledgling open-source detailed routers (based on OpenAccess) may become available. It is unlikely that open-source timing-driven routers with realistic timing models will emerge, nor will open-source routers that respect complex 45nm design rules.

Note that it makes little sense to implement a global router natively on a standard design database, since (1) the input to a global router cannot be taken directly from such a database, and (2) their output is mostly of use to detailed routers. This explains to some extent why the promise to port *BoxRouter* to OpenAccess in [11] has not been fulfilled in almost two years, and why the three current open-source routers rely heavily on their own internal data structures.

▷ **Contests.** For the physical design community to advance at its current rate, ISPD contests must continue. Due to the incorporation of runtime into the evaluation function (as well as overall progress in routing algorithms), winning routers at ISPD 2008 may run at least 4× faster or more than the winning routers at ISPD 2007, and produce better solutions on every benchmark.

However, as more tools become available through open-source initiatives, the competition will get tougher in future competitions, and the problems presented to participants will (hopefully) become more elaborate and realistic. By 2009, the "pure" global routing problem may well be considered by many to be solved, with seemingly little room on the table for further improvement.

▷ **Open problems** will shift to the incorporation of DFM concerns into global routing, integration with delay estimation, and coupling with placement. Detailed routing will remain unsolved and demand heavier infrastructure.

Appendix A: Diff Maps of Resource Utilization

Creative visualization techniques can shed light on otherwise invisible performance aspects of algorithms and tools. For example, Figure 2(a-c) shows congestion maps for the top three solutions to `adaptec1` from the contest. These were obtained by summing horizontal and vertical demand to create a single plot. While all three seem comparable, a simple variant of image subtraction (Figure 3(a-c)) reveals subtle differences between solutions. These trends can be used to explain biases in wirelength allocation. For instance, blue regions indicate areas where MaizeRouter tends to place more of its wires. Compared to FGR and *BoxRouter*, it prefers to route above blockages and close to chip edges. Green regions indicate high-usage areas for FGR, which include the center of the chip that are free of obstacles.

One important observation is that no single router dominates another over the entire design; hence, even though scalar metrics such as wirelength impose a total ordering over solutions, comparison at a finer level of detail can reveal more complex non-dominating relationships.

Acknowledgments

The authors wish to thank Gi-Joon Nam and the ISPD 2008 steering committee for the invitation to write this paper. Dr. M. D. Moffitt is supported by the 2007 Josef Raviv Memorial Postdoctoral Fellowship. J. A. Roy is supported by a Rackham Predoctoral Fellowship at the University of Michigan. Prof. I. L. Markov is partially supported by a grant from the National Science Foundation.

8. REFERENCES

[1] S. N. Adya, M. C. Yildiz, I. L. Markov, P. Villarrubia, P. N. Parakh, P. H. Madden, "Benchmarking for Large-Scale VLSI Placement and Beyond," in *TCAD*, vol. 23, no. 4, pp. 472–488, 2004.

[2] C. Albrecht, "Global routing by new approximation algorithms for multicommodity flow," in *TCAD*, vol. 20, no. 5, pp. 622–632, 2001.

[3] C. J. Alpert, C. Chu, and P. G. Villarrubia, "The coming of age of physical synthesis," in *Proc. of ICCAD 2007*, pp. 246–249.

[4] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Improved algorithms for hypergraph bipartitioning," in *Proc. of ASP-DAC 2006*, pp. 661–666.

[5] Z. Cao, T. Jing, J. Xiong, Y. Hu, L. He, and X. Hong, "DpRouter: A fast and accurate dynamic-pattern-based global routing algorithm," in *Proc. of ASP-DAC 2007*, pp. 256–261.

[6] R. C. Carden IV, J. Li, and C.-K. Cheng, "A global router with a theoretical bound on the optimal solution," in *TCAD*, vol. 15, no. 2, pp. 208–216, 1996.

[7] T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze, and M. Xie, "mPL6: enhanced multilevel mixed-size placement," in *Proc. of ISPD 2006*, pp. 212–214.

[8] H.-Y. Chen, M.-F. Chiang, Y.-W. Chang, L. Chen and B. Han, "Novel full-chip gridless routing considering double-via insertion," in *Proc. of DAC 2006*, pp. 755–760.

[9] T.-C. Chen, Y.-W. Chang and S.-C. Lin, "A novel framework for multilevel full-chip gridless routing," in *Proc. of ASP-DAC 2006*, pp. 636–641.

[10] M. Cho, K. Lu, K. Yuan, and D. Z. Pan, "BoxRouter 2.0: architecture and implementation of a hybrid and robust global bouter," in *Proc. of ICCAD 2007*, pp. 503–508.

[11] M. Cho and D. Z. Pan, "BoxRouter: A new global bouter based on box expansion and progressive ILP," in *Proc. of DAC 2006*, pp. 373–378.

[12] C. Chu, "FLUTE: fast lookup table based wirelength estimation technique," in *Proc. of ICCAD 2004*, pp. 696–701.

[13] C. C. N. Chu and Y.-C. Wong, "FLUTE: fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design," in *TCAD*, vol. 27, no. 1, pp. 70–83, 2008.

[14] J. Cong, M. Xie and Y. Zhang, "MARS – A multilevel full-chip gridless routing system," in *TCAD*, vol. 24, no. 3, pp. 382–394, 2005.

[15] J.-R. Gao, P.-C. Wu, and T.-C. Wang, "A new global router for modern designs," in *Proc. of ASP-DAC 2008*, pp. 232–237.

[16] R. Goering, "IC routing contest boosts CAD research," `www.eetimes.com/showArticle.jhtml?articleID=198500084`, 2007.

[17] R. T. Hadsell and P. H. Madden, "Improved global routing through congestion estimation," in *Proc. of DAC 2003*, pp. 28–31.

[18] M. Hanan, "On Steiner's problem with rectilinear distance," *SIAM Journal of Applied Mathematics*, vol. 14, pp. 255–265, 1966.

[19] J. Hu and S. S. Sapatnekar, "A survey on multi-net global routing for integrated circuits," *Integration, the VLSI Journal*, vol. 31, no. 1, pp. 1–49, 2001.

[20] A. Kahng, I. Mandoiu, and A. Zelikovsky, "Highly scalable algorithms for rectilinear and octilinear steiner trees," in *Proc. of ASP-DAC 2003*, pp. 827–833.

[21] A. Kahng, S. Reda, and Q. Wang, "Architecture and details of a high quality, large-scale analytical placer," in *Proc. of ICCAD 2005*, pp. 891–898.

[22] A. B. Kahng and X. Xu, "Accurate pseudo-constructive wirelength and congestion estimation," in *Proc. of SLIP 2003*, pp. 61–68.

[23] ——, "Pattern routing: use and theory for increasing predictability and avoiding coupling," in *TCAD*, vol. 21, no. 7, pp. 777–790, 2002.

[24] Z. Li, C. J. Alpert, S. T. Quay, S. S. Sapatnekar, and W. Shi, "Probabilistic congestion prediction with partial blockages," in *Proc. of ISQED 2007*, pp. 841–846.

[25] C.-W. Lin, S.-Y. Chen, C.-F. Li, Y.-W. Chang, and C.-L. Yang, "Efficient obstacle-avoiding rectilinear steiner tree construction," in *Proc. of ISPD 2007*, pp. 127–134.

[26] L. McMurchie and C. Ebeling, "Pathfinder: A negotiation-based performance-driven router for FPGAs," in *Proc. of FPGA 1995*, pp. 111–117.

[27] J. Lou, S. Krishnamoorthy, and H. S. Sheng, "Estimating routing congestion using probabilistic analysis," in *Proc. of ISPD 2001*, pp. 112–117.

[28] M. D. Moffitt, "MaizeRouter: Engineering an effective global router," in *Proc. of ASP-DAC 2008*, pp. 226–231.

[29] D. Müller, "Optimizing yield in global routing," in *Proc. of ICCAD 2006*, pp. 480–486.

[30] R. Nair, S. J. Hong, S. Liles, and R. Villani, "Global wiring on a wire-routing machine," in *Proc. of DAC 1982*, pp. 224–231.

[31] R. Nair, "A simple yet effective technique for global wiring," in *TCAD*, vol. 6, pp. 165–172, 1987.

[32] M. M. Ozdal and M. D. F. Wong, "Archer: a history-driven global routing algorithm," in *Proc. of ICCAD 2007*, pp. 488–495.

[33] M. Pan and C. Chu, "FastRoute: A step to integrate global routing into placement," in *Proc. of ICCAD 2006*, pp. 464–471.

[34] ——, "FastRoute 2.0: A high-quality and efficient global router," in *Proc. of ASP-DAC 2007*, pp. 250–255.

[35] ——, "IPR: An integrated placement and routing algorithm," in *Proc. of DAC 2007*, pp. 59–62.

[36] J. A. Roy and I. L. Markov, "High-performance routing at the nanometer scale," in *Proc. of ICCAD 2007*, pp. 496–502.

[37] L. K. Scheffer, L. Lavagno and G. Martin, eds., *Electronic Design Automation for Integrated Circuits Handbook*, Chapter 8: Routing, CRC Press, 2006.

[38] C.-W. Sham and E. F. Y. Young, "Congestion prediction in early stages," in *Proc. of SLIP 2005*, pp. 91–98.

[39] J. Westra, C. Bartels, and P. Groeneveld, "Probabilistic congestion prediction," in *Proc. of ISPD 2004*, pp. 204–209.

[40] J. Westra and P. Groeneveld, "Is probabilistic congestion estimation worthwhile?" in *Proc. of SLIP 2005*, pp. 99–106.

[41] Y. Yang, T. Jing, X. Hong, Y. Hu, Q. Zhu, X. Hu, and G. Yan, "Via-aware global routing for good VLSI manufacturability and high yield" in *Proc. of ASAP 2005*, pp. 198–203.

[42] M. Zachariasen, "Rectilinear Full Steiner Tree Generation," in *Networks*, vol. 33, pp. 125–143, 1999.

[43] `www.ispd.cc/ispd07_contest.html`

[44] `www.ece.ucsb.edu/~kastner/labyrinth/`

[45] `vlsicad.ucsd.edu/GSRC/bookshelf/Slots/RSMT/FastSteiner/`

[46] `home.eng.iastate.edu/~cnchu/flute.html`

[47] `www.cerc.utexas.edu/~thyeros/boxrouter/boxrouter.htm`

[48] `www.cerc.utexas.edu/utda/download/BoxRouter.htm`

[49] `vlsicad.eecs.umich.edu/BK/FGR/`

[50] `www.eecs.umich.edu/~mmoffitt/MaizeRouter/`