Building Graph-based Programming Strategies for Reconfigurable Photonic Circuits

Xiangfeng Chen and Wim Bogaerts

Ghent University - IMEC, Photonic Research Group,
Department of Information Technology, Gent, Belgium
Center for Nano and Biophotonics, Ghent University, Gent, Belgium.

We abstract optical waveguide meshes into graph representations to solve practical light distribution problems, leveraging structures and algorithms from graph theory. One of the substantial problems is single input-output signal routing inside the meshes. Whilst respecting physical restrictions regarding the directional flow of light, three types of graph representations are compared in terms of feasibility for solving this problem. Of the three types, the directed graph with eight artificial nodes proves to be superior. A representation of this kind takes us one step closer to a fully programmable optical core with complex functionalities, especially in photonic meshes with feedback paths.

Introduction

Large-scale programmable photonic circuits with optical feedback paths allow us to realize a wide range of functionalities [1], but they lack automated design and programming strategies. Our previous work proposed a graph-based perspective for the design and programming of photonic integrated circuits with feedback architectures [2]. Respecting physical restrictions regarding the directional flow of light, a customized Dijkstra's Algorithm was implemented. This way, we reduce the single input and output signal routing problems to the shortest pathfinding problems in the graph which are basic but essential light distribution problems in our programmable mesh. However, scrutiny on such graph representation is needed. The accessibility of the algorithm solutions for each optical port of the mesh circuit has been left unverified. In this paper, we expose this problem, and identify two types of violation cases. Based on these violation cases, we provide evaluations among three graph presentations.

Graph Representations and Evaluations

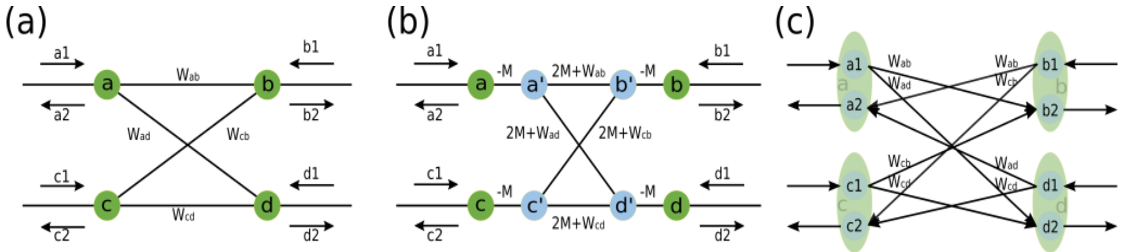

Figure 1. (a) original graph for a 2 x 2 coupler in the mesh; (b) an alternative auxiliary graph with 4 extra nodes a', b', c', d' and artificial edge weights; (c) an auxiliary graph where every port a, b, c, d is represented by an in and out node a1, a2, b1, b2 …

For the initial graph representation that we proposed in previous work [2], the ports of the photonic building blocks are abstracted as nodes in the photonic graph. Two-port building blocks are being represented by two nodes sharing a single edge. Four-port couplers shown in Fig.1(a) have 4 ports, thus 4 nodes in graph representation, named a, b, c, and d. One of the possible implementations for the hexagonal mesh consists of repeatable unit cells that have 3 programmable 2 × 2 couplers with 3 phase shifters to interconnect them. We map four different graph representations with three coupler schemes in Fig.1 with this repeatable unit cell highlighted in color boxes in Fig. 2. We assign specific edge attributes to each connection such as insertion loss, power consumption, and cross-talk. In our graph abstraction, we model different propagation loss as weights for different connections (10 for "cross", 5 for "bar", 4 for phase shifters and 1 for all waveguide connections). These four graph representations proposed in Fig.2 are compared for the single input-output light distribution which is reduced to the single pair shortest pathfinding problem. The result of the routing will be an optimized solution on the total propagation loss. However, there are two types of physical violations when we route single pair shortest path in the initial graph representation of our circuit: Type 1, light abruptly changes direction inside a coupler (mixing "bar" and "cross" states in one coupler); Type 2, some nodes are not reachable by the algorithm while they should be accessible in the actual mesh. We try to solve them through three perspectives: algorithms, weights, and edge directions. Our customized Dijkstra's Algorithm does solve Type 1 violation [2] but fails to address the Type 2 violation which is the problem of accessibility as shown in Fig. 3(a).

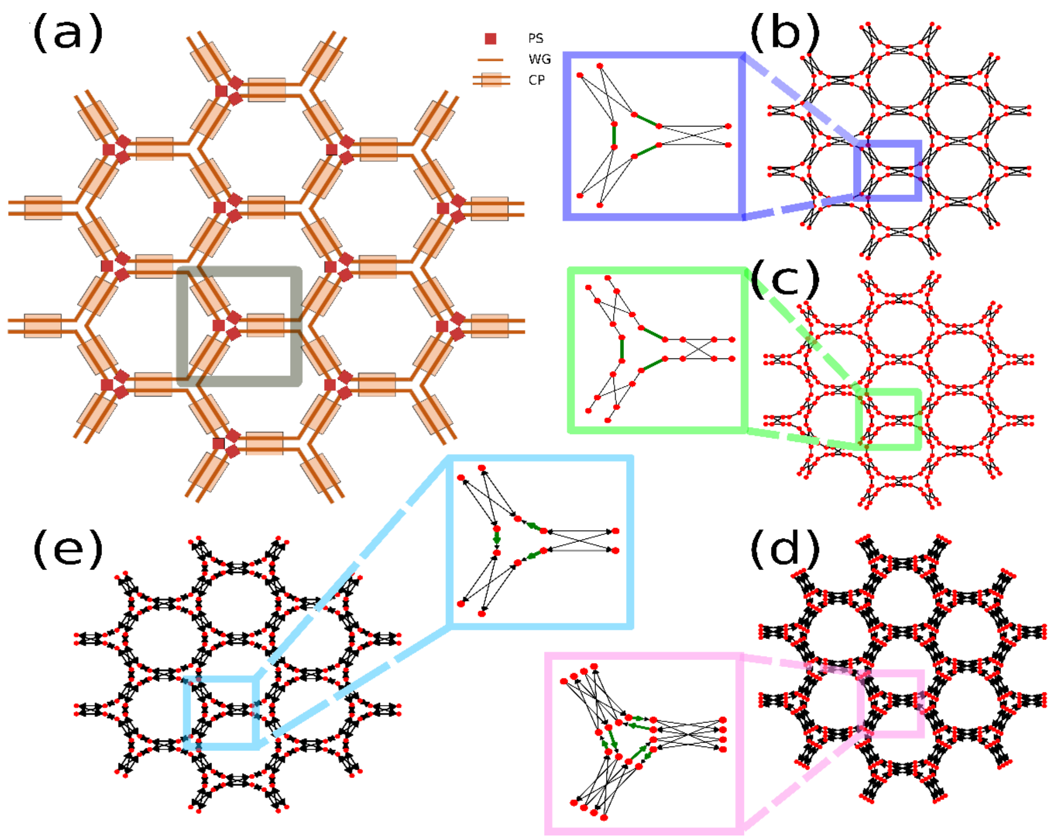

Figure 2. (a) Schematic drawing of our example hexagonal waveguide mesh; (b) original undirected graph representation as in Fig.1(a); (c) undirected graph with negative weights based on coupler implementation in Fig.1(b); (d) directed graph utilizes the coupler scheme in Fig.1(c); (e) The same graph as (d) but projecting nodes representing the same ports on the same location.

Many algorithms support weights that can be positive and negative. To solve Type 1 violations we extend out graph with four more artificial nodes and assign a weight M as an arbitrarily large number. This way, the weighting scheme in Fig.1(b) would favor the connections in the same coupling state for every coupler while the algorithm is trying to find the shortest path. The huge penalty M will prevent those internal paths from mixing both coupling states, which solves the Type 1 violation. However, it shares the same intrinsic problem as an undirected graph as the coupler implementation in Fig.1(a). Dijkstra's Algorithm has a spanning searching tree for the shortest path. If one of the nodes in the current searching branches is being been looked into, it gets compared in the priority queue [3]. It will be marked as "visited" and future searching branches are forbidden to consider the node again. The examples of such searching branches are simplistically expressed in blue and yellow alternative routes in Fig. 3(a) and 3(b). Since we arbitrarily assigned the weight of "cross" edges larger than "bar" edges as we are abstracting our photonic graph. The blue path cost less than yellow from "s" to "m" thus blue path has taken "m" and marked it as "visited". For the graph in Fig. 3(a), the yellow path cannot absorb "m" as a node in the path and reach "t" through "m". For the graph in Fig. 3(b), having mixing states of the coupler highlighted in red is the only possible route left for blue path reaching "t", even though the cost of the yellow route continued with an edge from "m" to "t" are much cheaper than the cost of the blue counterpart. This resulted in the red dots being unreachable for the first graph and unphysical for the latter one.

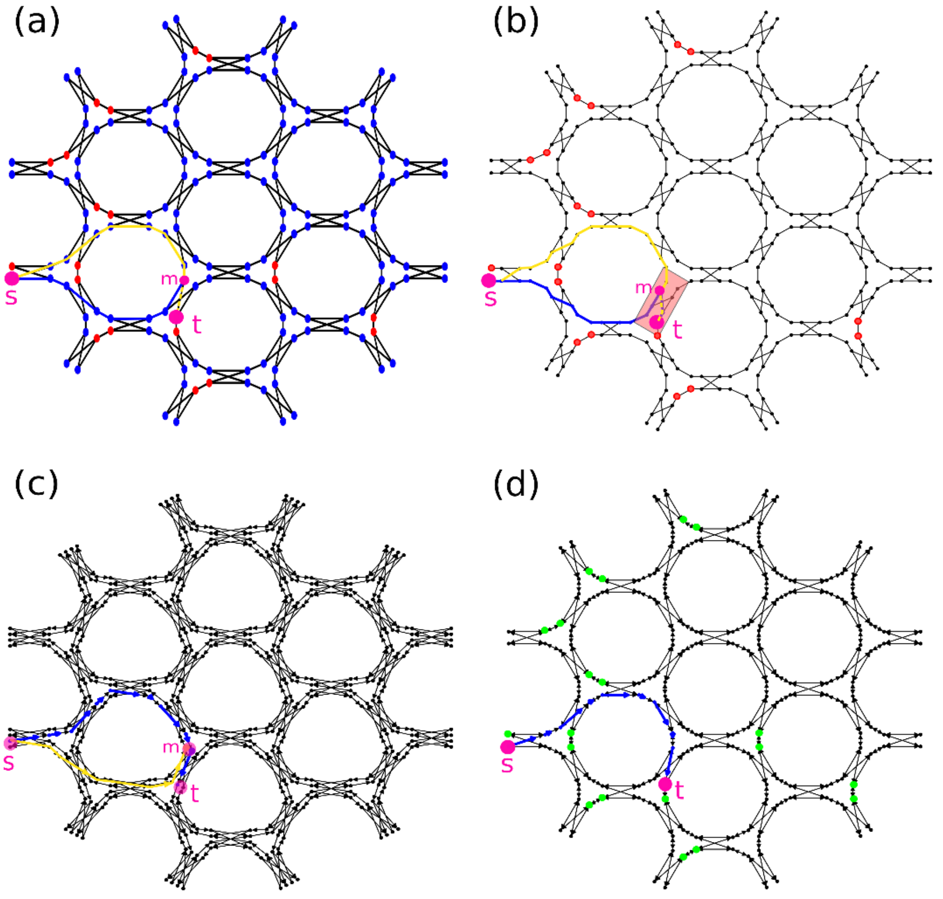

Figure 3. The red dots in (a) and (b) show unreachable and nonphysical destinations, respectfully, for signal routing starting from the source node (annotated in "s"), through the middle node ("m"), to the terminal node ("t"). The blue route is the solution that these algorithms provide. While the yellow paths in (a), (b) and (c) are possible alternatives to these blue paths. (d) is the graph in Fig.2(e) with green dots showing its valid accessibility.

Graphs can be undirected and directed. To fully represent our circuit, we proposed coupler implementation in Fig.1(c) to translate the possible light flow of the same port into two nodes. The directed coupler implementation naturally solves the Type 1 violation. The light comes from a1 only goes to b2 and d2. For the Type 2 violation, this graph representation allows Dijkstra's algorithm to compare the cost of routes passing each port with both directions as opposed to the problem caused by undirected counterparts. However, constraints are needed to penalize edges that two nodes representing one port are both taken by the shortest path. We use heuristic programming to update new edge costs with the sum of the original weights and the products of the history of violations and the user-defined violation cost. [4] Fig.3(c) shows correct routing results compared with the undirected counterpart in Fig.3(a) and Fig.3(b). The cost of possible routes after getting through node "n" in both yellow and blue are evaluated in the priority queue of the searching tree of the standard Dijkstra algorithm. For better user visualization, we overlay the two nodes representing the same port at the same location and show it in Fig.2(e) and Fig.3(d).

Conclusions

2×2 tunable couplers play an essential role in our graph model because these couplers are key components to distribute light inside meshes. We proposed three coupler implementations then translated them into graph representations with evaluations in terms of physical restrictions and valid solutions for our circuits. The last directed graph is superior to fully represent our circuit in the graph. The application of graph theory to this new field would help us make a solid improvement towards systematically controlled programable meshes.

Acknowledgements

This work has received funding from the European Research Council through grant agreement No 725555 (PhotonicSWARM).

References

[1] W. Bogaerts, D.A.B. Miller, J. Capmany, "The New World of Programmable Photonics", IEEE Photonics Society Summer Topical Meeting Series (SUM), Ft. Lauderdale, FL, USA, 2019, p.ME1.1.

[2] X. Chen and W. Bogaerts, "A Graph-based Design and Programming Strategy for Reconfigurable Photonic Circuits," IEEE Photonics Society Summer Topical Meeting Series (SUM), Ft. Lauderdale, FL, USA, 2019, p.ME2.2.

[3] D. E. Knuth, "A generalization of dijkstra's algorithm," Information Processing Letters, vol. 6, no. 1, pp. 1–5, 1977.

[4] L. McMurchie and C. Ebeling, "Pathfinder. A Negotiation-Based Performance-Driven Router for FPGAs." in Reconfigurable Computing, 2008.