An Efficient Branch-and-Bound Routing Algorithm for Optical NoCs

Yihao Liu, Yaoyao Ye

Shanghai Jiao Tong University, China

yeyaoyao@sjtu.edu.cn

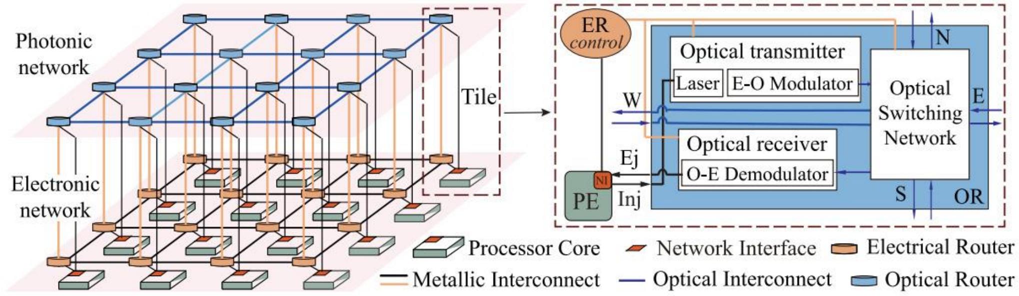
Outline

- **Research background**
- **Branch-and-bound routing for ONoCs**
  - Branch-and-bound routing pruned at turning nodes
  - Variant-1: Branch-and-bound thermal-aware routing
  - Variant-2: Branch-and-bound bi-objective routing
- **Simulation results**
- **Conclusion and future works**

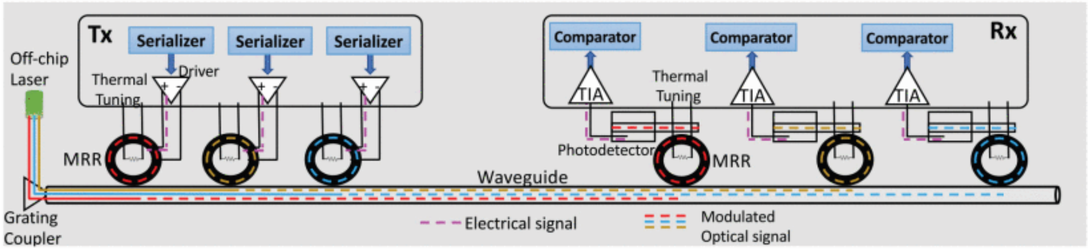
Optical NoC Architecture

- **Advantages of ONoCs**
  - High bandwidth
  - High energy efficiency

- **Challenges of ONoCs**
  - Temperature sensitivity
  - Optical link contention



Mesh based photo-electric hybrid NoC [1]



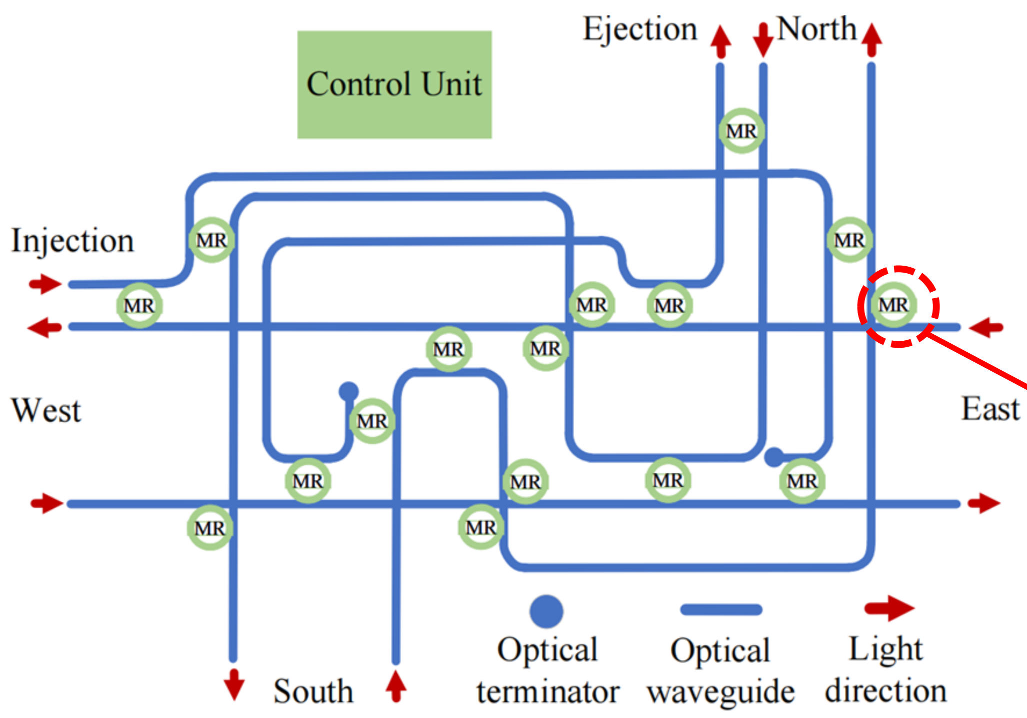
Wavelength division multiplexing [2]

[1] M. Li, W. Liu, L. H. Duong, P. Chen, L. Yang, and C. Xiao, “Contention-aware routing for thermal-reliable optical networks-on-chip,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 40, no. 2, pp. 260–273, 2020.

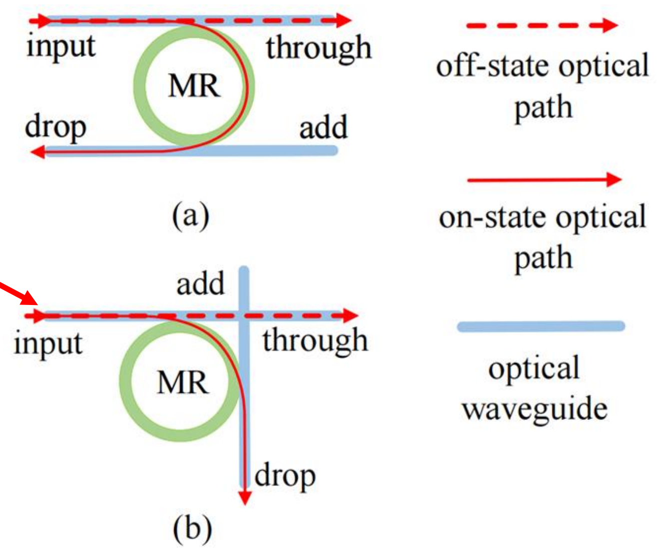
[2] A. Narayan, Y. Thonnart, P. Vivet and A. K. Coskun, "PROWAVES: Proactive Runtime Wavelength Selection for Energy-Efficient Photonic NoCs," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 40, no. 10, pp. 2156-2169, Oct. 2021

Challenges of ONoCs

- **Temperature sensitivity**
  - Silicon microresonator (MR) as optical switching element
  - Chip temperature fluctuates in time and space
  - Resonant wavelength of MR does not match laser wavelength



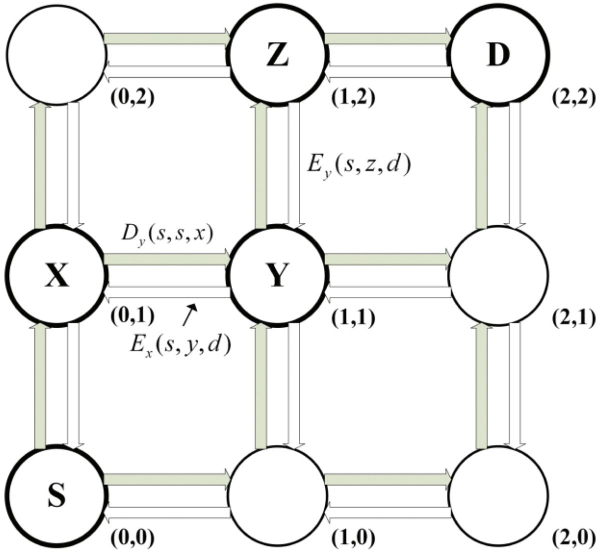
Optical router



Optical switching element

Challenges of ONoCs

- **Thermal-aware adaptive routing [1, 2]**
  - Circumvent high-temperature regions for routing
  - Congestion exacerbates within the low-temperature regions on the chip

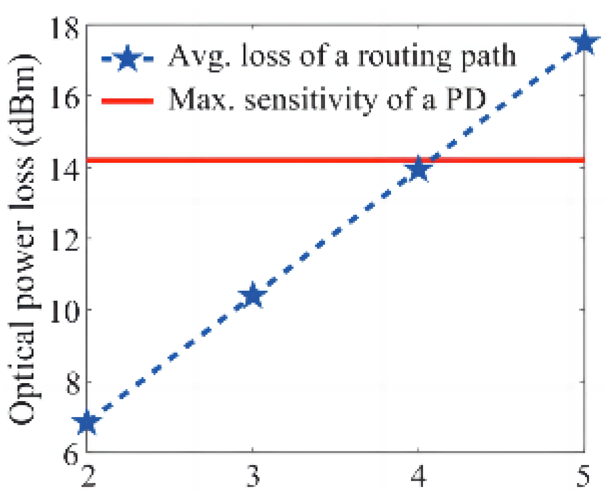


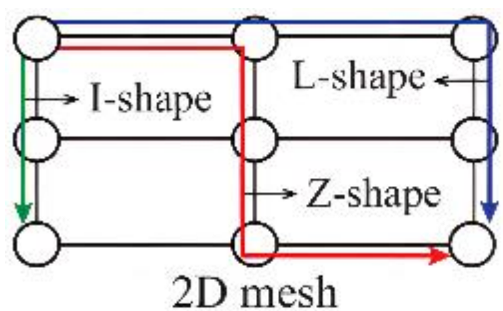
[1] Z. Zhang and Y. Ye, "A learning-based thermal-sensitive power optimization approach for optical NoCs," ACM Journal on Emerging Technologies in Computing Systems (JETC), 2018, 14(2): 21

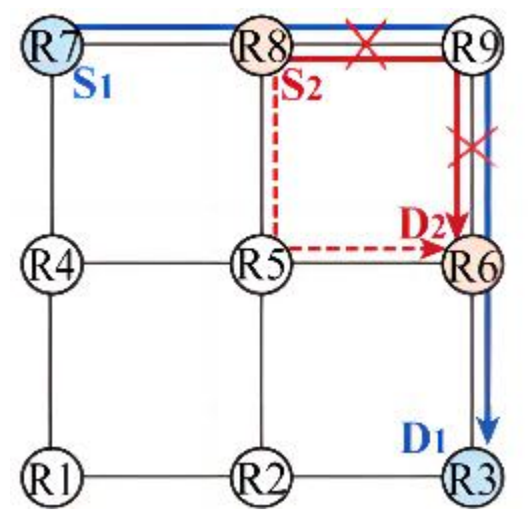
[2] W. Zhang and Y. Ye, "A table-free approximate Q-learning-based thermal-aware adaptive routing for optical NoCs," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 40, no. 1, pp. 199–203, 2021.

Challenges of ONoCs

- **Heuristic contention-aware centralized routing [1]**
  - The scope of available paths is too small
  - The optical power loss optimization is less efficient
  - Centralized routing matches circuit switching mechanisms of ONoCs



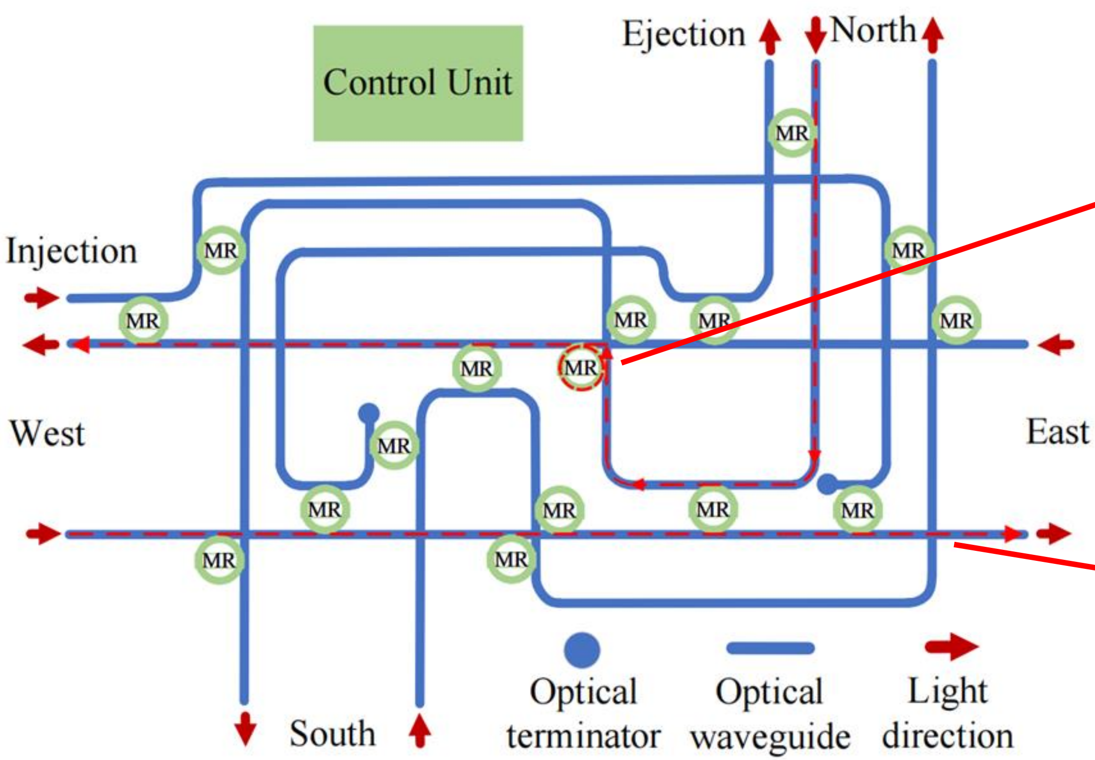




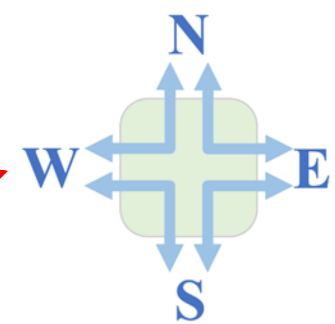
[1] M. Li, W. Liu, L. H. Duong, P. Chen, L. Yang, and C. Xiao, "Contention-aware routing for thermal-reliable optical networks-on-chip," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 40, no. 2, pp. 260–273, 2020.

Branch-and-bound routing pruned at turning nodes (BBR)

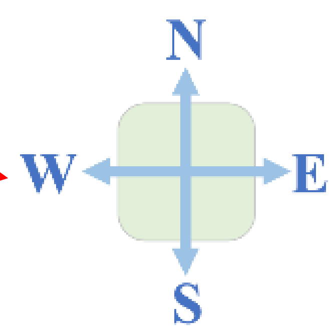
- **Characteristics of optical power loss**
  - Thermal-induced optical power loss primarily originates from active switches
  - Paths meeting thermal reliability contain fewer turning nodes



Cygnus optical router



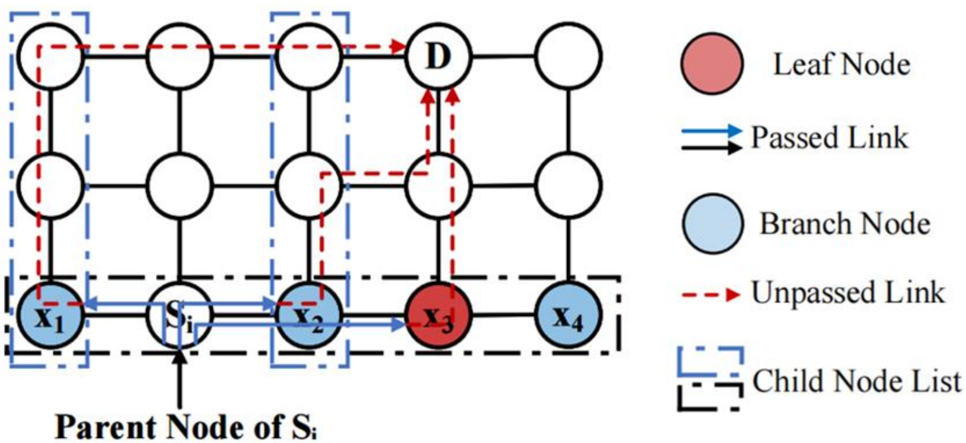
Turning node



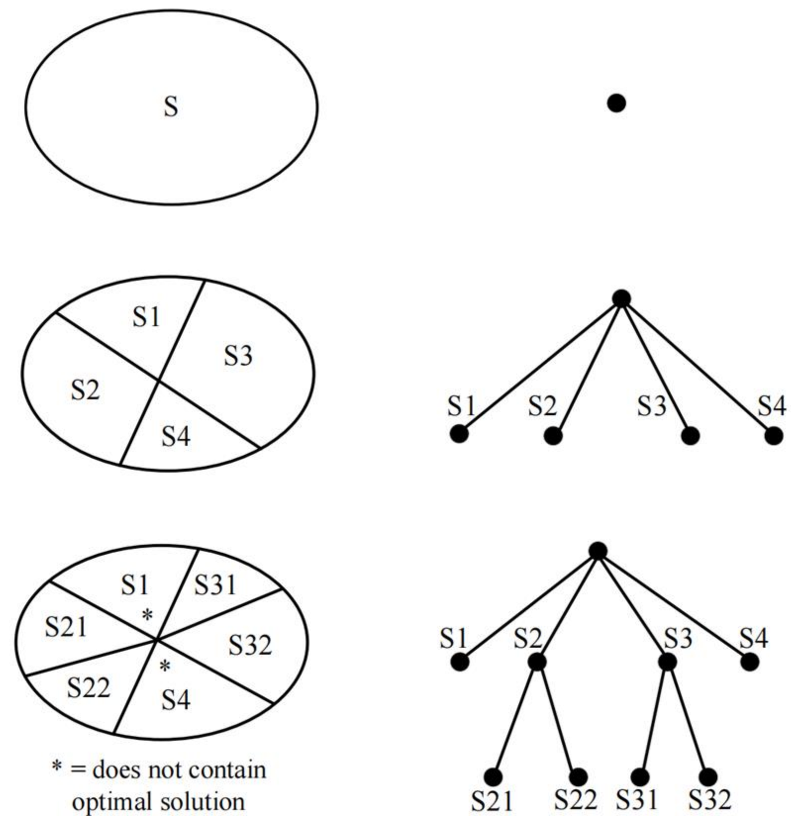
Through node

Branch-and-bound routing pruned at turning nodes (BBR)

- **Branch-and-Bound (B&B)**
  - Using a tree search strategy to implicitly enumerate all possible solutions
- **Components of B&B**
  - Branching strategy
  - Search strategy
  - Pruning rules



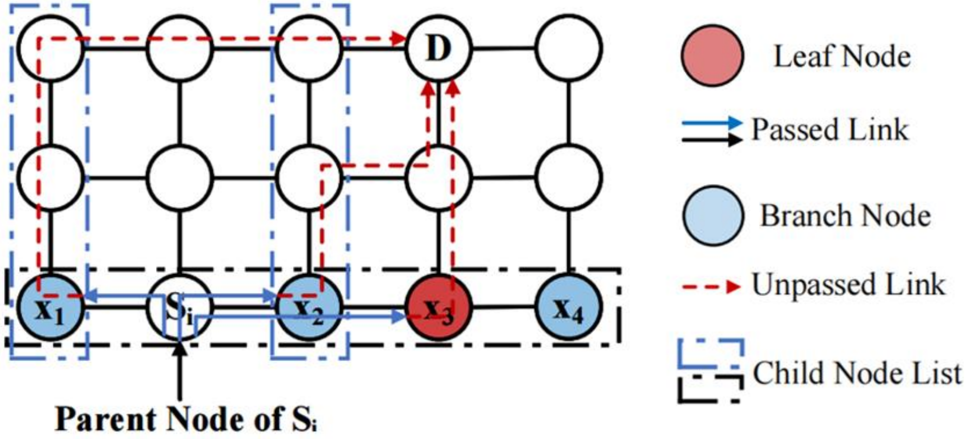
Process of one BBR iteration



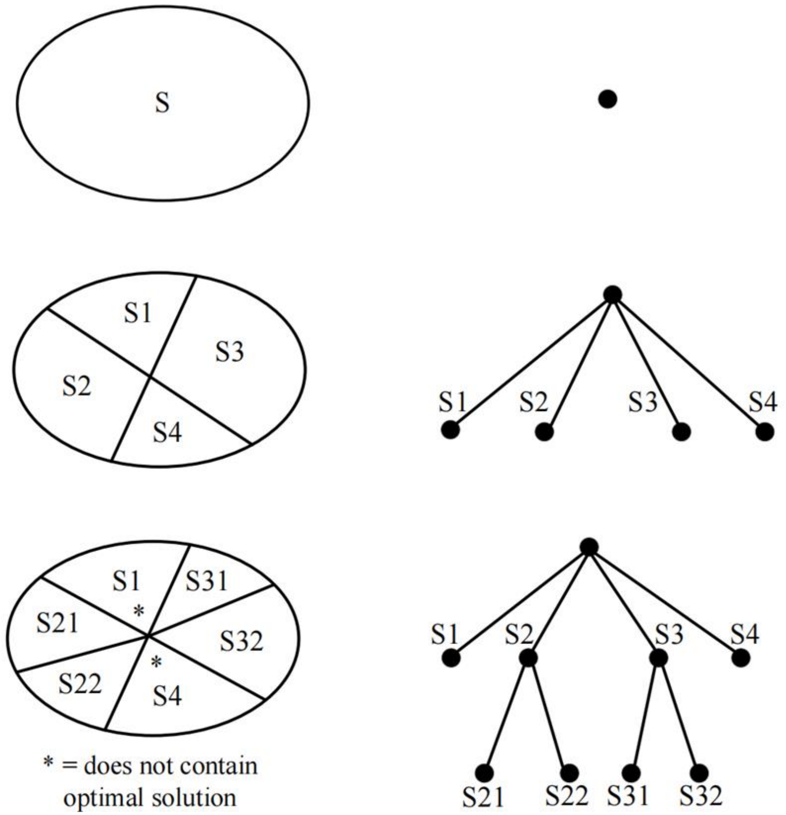
Process of Branch-and-Bound

Branch-and-bound routing pruned at turning nodes (BBR)

- **Turning node based branching strategy**
- **Turning node based bounding function**
  - $V_x = V_{actual_x} + V_{min_{x \to D}}$
    $= V_{actual_i} + V_{actual_{i \to x}} + V_{min_{x \to D}}$
  - $V_x < Upper_V$

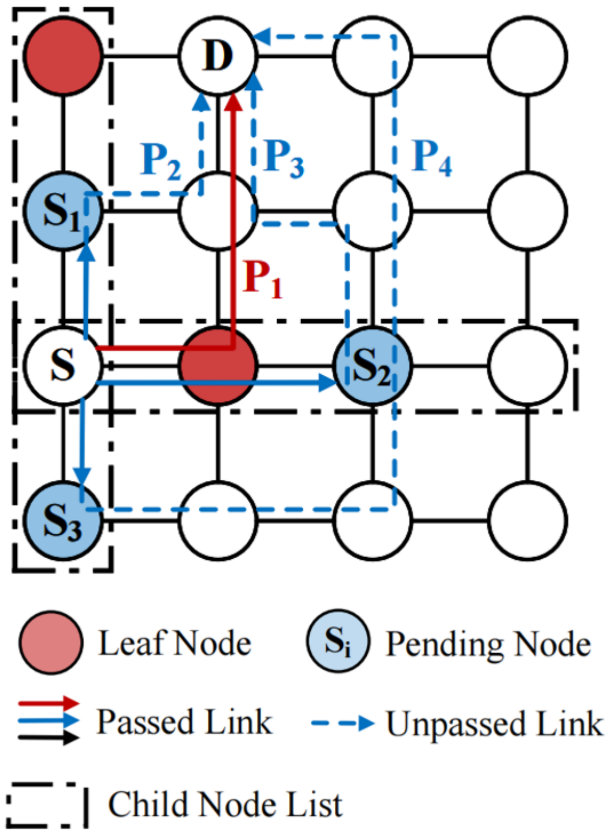


Process of one BBR iteration



Process of Branch-and-Bound

Branch-and-bound routing pruned at turning nodes (BBR)



Algorithm 1: Branch-and-Bound Routing (BBR)

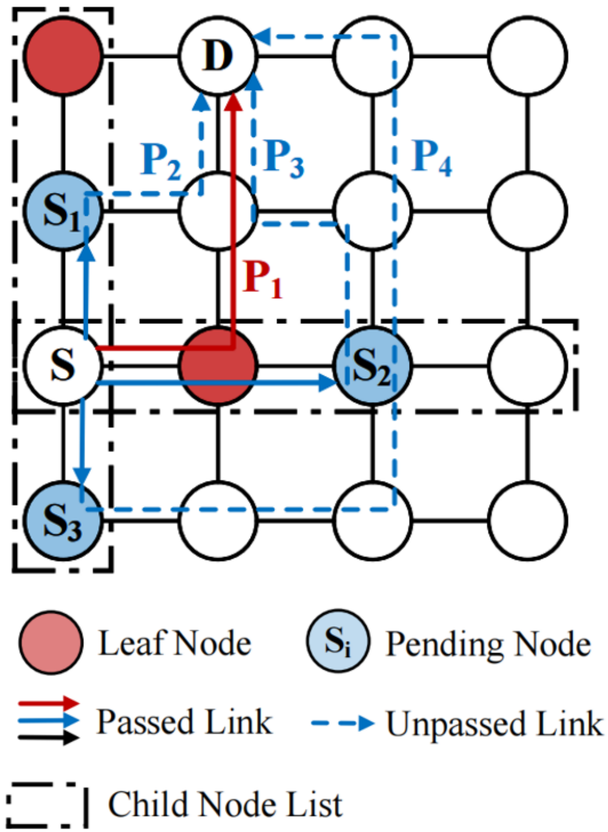
input : global information, source S, destination D, initial upper bound V_{init}, initial optimal path P_{init}

output: the optimal path P_{best}

```
PT ← {S}, [Lower, Upper] ← [0, V_init],
    P_best ← P_init // initialization
while PT ≠ ∅ do
    S_i ← pop node i with minimum value from PT
    for x ∈ child node list of S_i do
        V_x ← GetBoundFuncValue(V_i, x)
        Flag_x ← CheckPrunFlag()
        if Flag_x then
            if x is branch node then
                add node x to PT
            else if x is leaf node then
                P_best ← P_i + {x, D}, Upper ← V_x
                delete nodes in PT where V > Upper
```

	Pending node	Best path	Pending table
Iteration$_1$		P_{init}	S
Iteration$_2$			

Branch-and-bound routing pruned at turning nodes (BBR)



Algorithm 1: Branch-and-Bound Routing (BBR)

input : global information, source S, destination D, initial upper bound V_{init}, initial optimal path P_{init}

output: the optimal path P_{best}

$PT \leftarrow \{S\}$, $[Lower, Upper] \leftarrow [0, V_{init}]$, $P_{best} \leftarrow P_{init}$ `// initialization`

while $PT \neq \varnothing$ **do**

    $S_i \leftarrow$ pop node i with minimum value from PT

    for $x \in$ *child node list of* S_i **do**

        $V_x \leftarrow$ `GetBoundFuncValue`(V_i, x)

        $Flag_x \leftarrow$ `CheckPrunFlag()`

        if $Flag_x$ **then**

            if x *is branch node* **then**

                add node x to PT

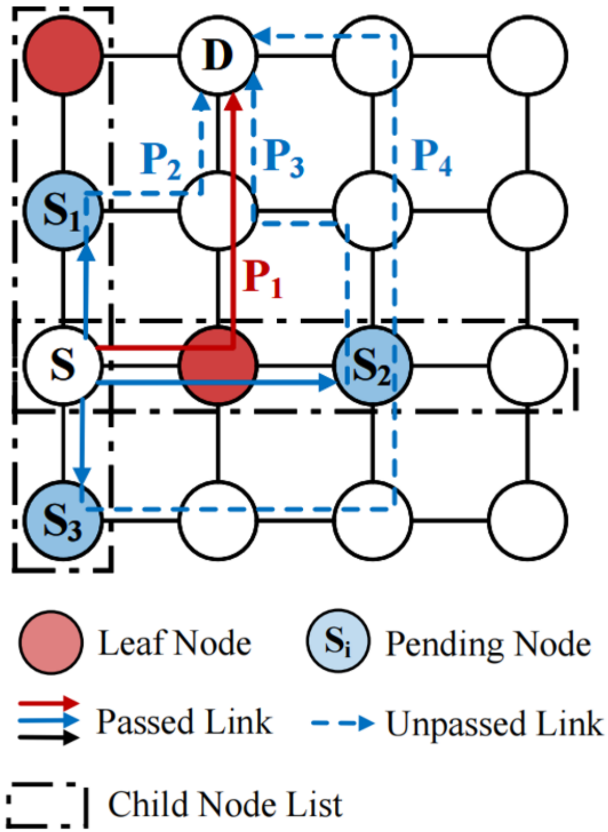
            else if x *is leaf node* **then**

                $P_{best} \leftarrow P_i + \{x, D\}$, $Upper \leftarrow V_x$

                delete nodes in PT where $V > Upper$

	Pending node	Best path	Pending table
Iteration$_1$	S	P_{init}	
Iteration$_2$			

Branch-and-bound routing pruned at turning nodes (BBR)



Algorithm 1: Branch-and-Bound Routing (BBR)

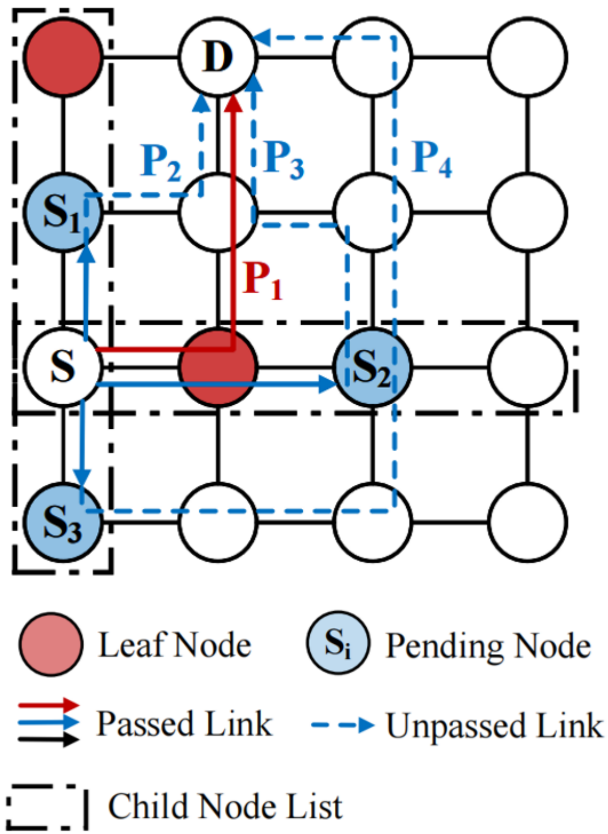
input : global information, source S, destination D, initial upper bound V_{init}, initial optimal path P_{init}

output: the optimal path P_{best}

$PT \leftarrow \{S\}$, $[Lower, Upper] \leftarrow [0, V_{init}]$,
  $P_{best} \leftarrow P_{init}$ `// initialization`
while $PT \neq \varnothing$ **do**
  $S_i \leftarrow$ pop node i with minimum value from PT
  for $x \in$ *child node list of* S_i **do**
    $V_x \leftarrow$ `GetBoundFuncValue`(V_i, x)
    $Flag_x \leftarrow$ `CheckPrunFlag()`
    if $Flag_x$ **then**
      if *x is branch node* **then**
        add node x to PT
      else if *x is leaf node* **then**
        $P_{best} \leftarrow P_i + \{x, D\}$, $Upper \leftarrow V_x$
        delete nodes in PT where $V > Upper$

	Pending node	Best path	Pending table
Iteration$_1$	S	P_1	
Iteration$_2$			

Branch-and-bound routing pruned at turning nodes (BBR)



Algorithm 1: Branch-and-Bound Routing (BBR)

input : global information, source S, destination D, initial upper bound V_{init}, initial optimal path P_{init}

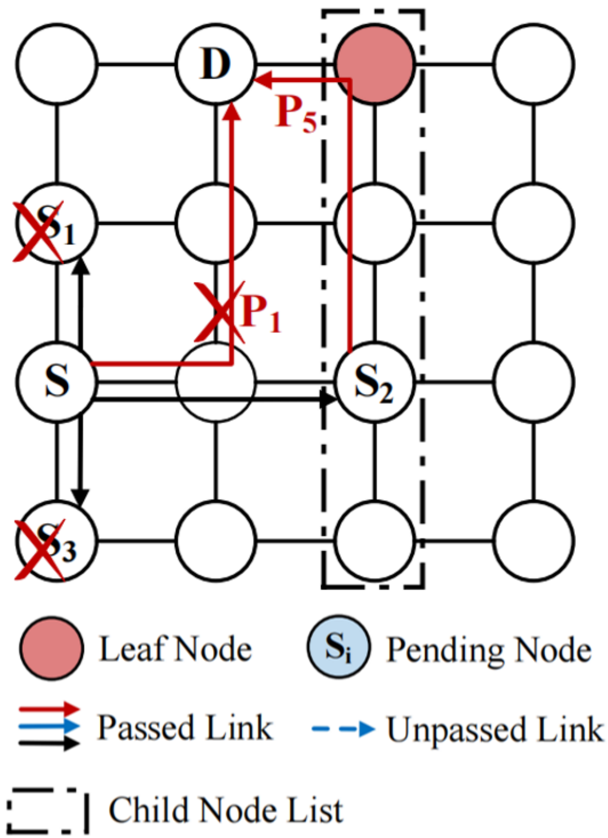
output: the optimal path P_{best}

$PT \leftarrow \{S\}$, $[Lower, Upper] \leftarrow [0, V_{init}]$, $P_{best} \leftarrow P_{init}$ `// initialization`

while $PT \neq \varnothing$ **do**
- $S_i \leftarrow$ pop node i with minimum value from PT
- **for** $x \in$ *child node list of* S_i **do**
  - $V_x \leftarrow$ `GetBoundFuncValue`(V_i, x)
  - $Flag_x \leftarrow$ `CheckPrunFlag()`
  - **if** $Flag_x$ **then**
    - **if** x *is branch node* **then**
      - add node x to PT
    - **else if** x *is leaf node* **then**
      - $P_{best} \leftarrow P_i + \{x, D\}$, $Upper \leftarrow V_x$
      - delete nodes in PT where $V > Upper$

	Pending node	Best path	Pending table
Iteration$_1$	S	P_1	S_1, S_2, S_3
Iteration$_2$			

Branch-and-bound routing pruned at turning nodes (BBR)



Algorithm 1: Branch-and-Bound Routing (BBR)

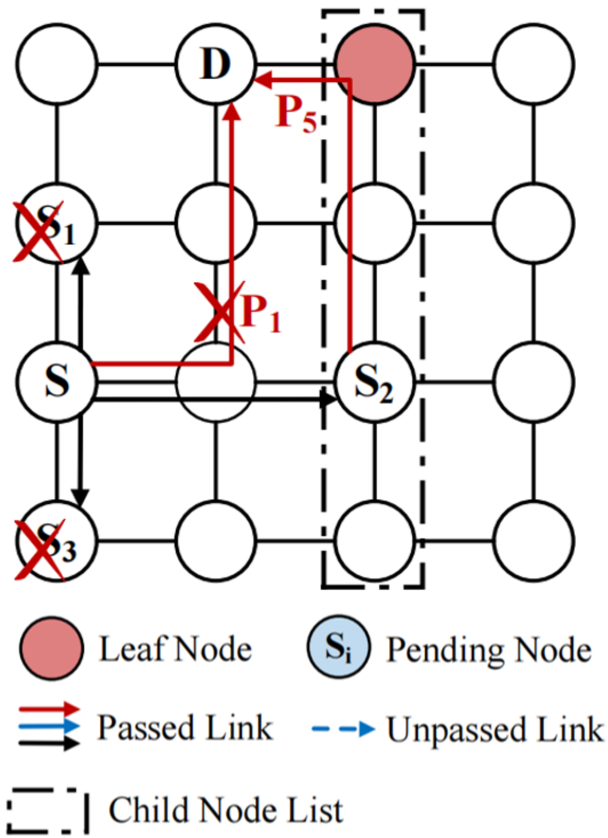
input : global information, source S, destination D, initial upper bound V_{init}, initial optimal path P_{init}

output: the optimal path P_{best}

$PT \leftarrow \{S\}$, $[Lower, Upper] \leftarrow [0, V_{init}]$,
  $P_{best} \leftarrow P_{init}$ `// initialization`
while $PT \neq \varnothing$ **do**
  $S_i \leftarrow$ pop node i with minimum value from PT
  for $x \in$ *child node list of* S_i **do**
    $V_x \leftarrow$ `GetBoundFuncValue`(V_i, x)
    $Flag_x \leftarrow$ `CheckPrunFlag()`
    if $Flag_x$ **then**
      if x *is branch node* **then**
        add node x to PT
      else if x *is leaf node* **then**
        $P_{best} \leftarrow P_i + \{x, D\}$, $Upper \leftarrow V_x$
        delete nodes in PT where $V > Upper$

	Pending node	Best path	Pending table
Iteration$_1$	S	P_1	S_1, S_2, S_3
Iteration$_2$	S_2	P_1	S_1, S_3

Branch-and-bound routing pruned at turning nodes (BBR)



Algorithm 1: Branch-and-Bound Routing (BBR)

input : global information, source S, destination D, initial upper bound V_{init}, initial optimal path P_{init}

output: the optimal path P_{best}

$PT \leftarrow \{S\}$, $[Lower, Upper] \leftarrow [0, V_{init}]$, $P_{best} \leftarrow P_{init}$ `// initialization`

while $PT \neq \varnothing$ **do**

  $S_i \leftarrow$ pop node i with minimum value from PT

  for $x \in$ *child node list of* S_i **do**

    $V_x \leftarrow$ `GetBoundFuncValue`(V_i, x)

    $Flag_x \leftarrow$ `CheckPrunFlag()`

    if $Flag_x$ **then**

      if x *is branch node* **then**

        add node x to PT

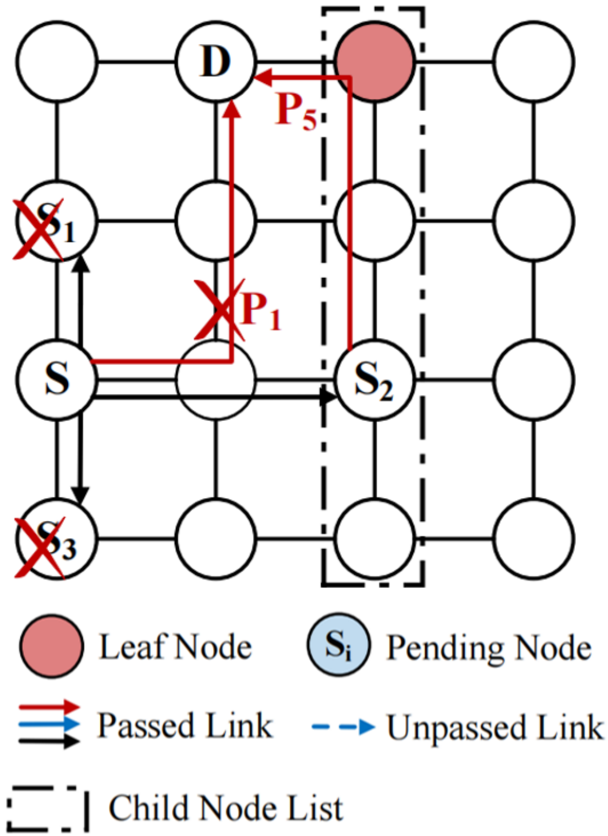
      else if x *is leaf node* **then**

        $P_{best} \leftarrow P_i + \{x, D\}$, $Upper \leftarrow V_x$

        delete nodes in PT where $V > Upper$

	Pending node	Best path	Pending table
Iteration$_1$	S	P_1	S_1, S_2, S_3
Iteration$_2$	S_2	P_5	~~S_1, S_3~~

Branch-and-bound routing pruned at turning nodes (BBR)



Algorithm 1: Branch-and-Bound Routing (BBR)

input : global information, source S, destination D, initial upper bound V_{init}, initial optimal path P_{init}

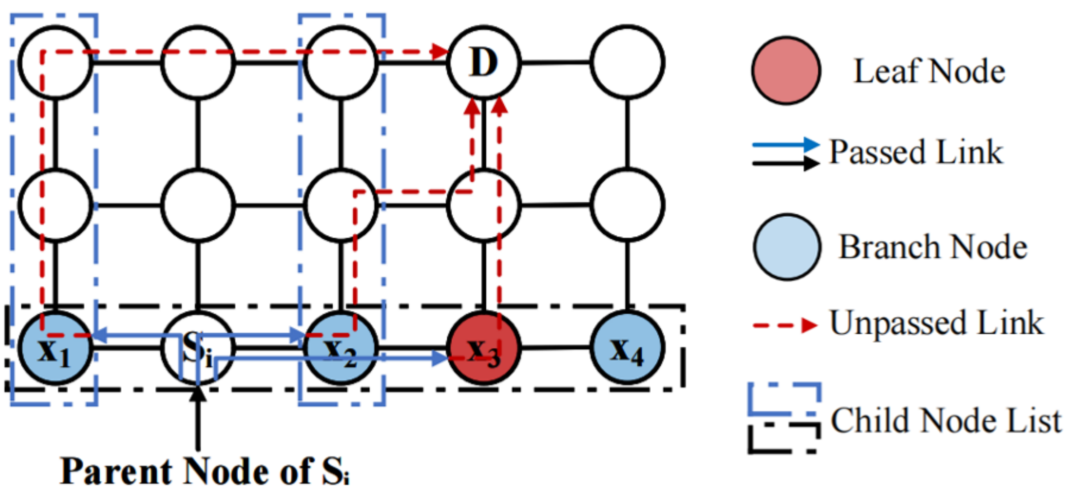
output: the optimal path P_{best}

$PT \leftarrow \{S\}$, $[Lower, Upper] \leftarrow [0, V_{init}]$,
  $P_{best} \leftarrow P_{init}$ `// initialization`
while $PT \neq \varnothing$ **do**
  $S_i \leftarrow$ pop node i with minimum value from PT
  for $x \in$ *child node list of* S_i **do**
    $V_x \leftarrow$ `GetBoundFuncValue`(V_i, x)
    $Flag_x \leftarrow$ `CheckPrunFlag()`
    if $Flag_x$ **then**
      if x *is branch node* **then**
        add node x to PT
      else if x *is leaf node* **then**
        $P_{best} \leftarrow P_i + \{x, D\}$, $Upper \leftarrow V_x$
        delete nodes in PT where $V > Upper$

	Pending node	Best path	Pending table
Iteration$_1$	S	P_1	S_1, S_2, S_3
Iteration$_2$	S_2	P_5	

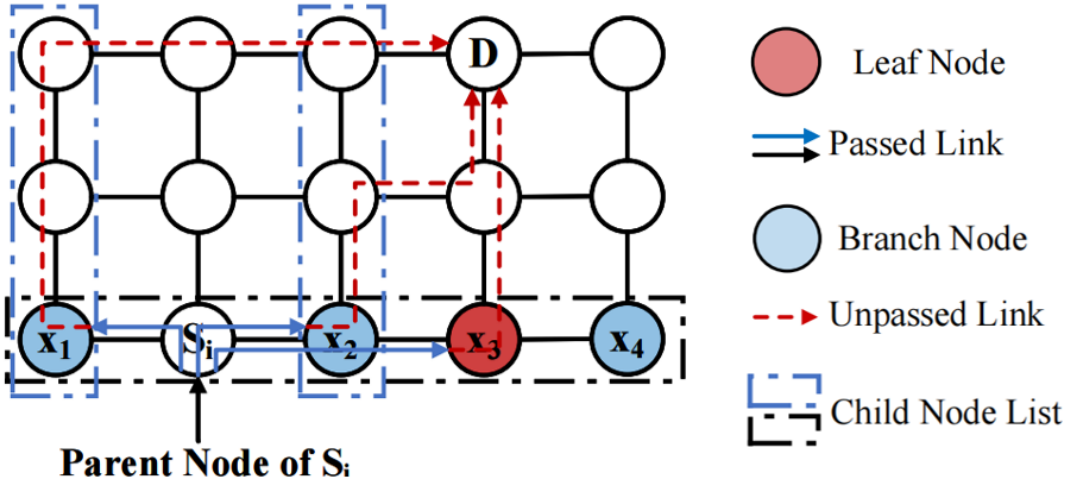
Variant-1: Branch-and-bound thermal-aware routing (BBTR)

- **Additional optical power loss of paths**
  - L_{active_i} is optical power loss of turning node S_i
  - $\Delta L = \sum_{i=1}^{k}\left(L_{active_i} + \Delta h_i \cdot L_{wire}\right)$
- **Bounding function**
  - $\Delta L_x = \Delta L_{actual_i} + \Delta L_{actual_{i\to x}} + \Delta L_{min_{x\to D}}$
- **Pruning rules**
  - $\Delta L_x < Upper_{\Delta L}$



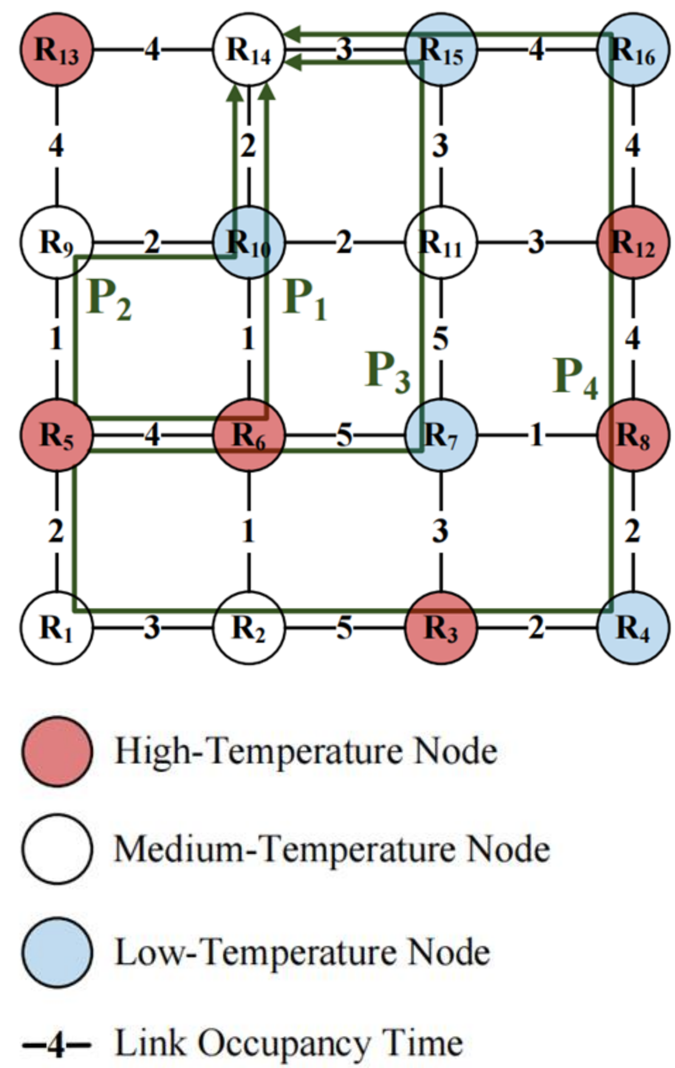
Variant-1: Branch-and-bound thermal-aware routing (BBTR)

- **Additional optical power loss of paths**
  - L_{active_i} is optical power loss of turning node S_i
  - $\Delta L = \sum_{i=1}^{k}\left(L_{active_i} + \Delta h_i \cdot L_{wire}\right)$
- **Bounding function**
  - $\Delta L_x = \Delta L_{actual_i} + \Delta L_{actual_{i \to x}} + \Delta L_{min_{x \to D}}$
- **Pruning rules**
  - $\Delta L_x < Upper_{\Delta L}$



Variant-2: Branch-and-bound bi-objective routing (3BOR)

- **Link occupancy model**
  - $t_{max} = max(t_i), i \in P_{link}$
  - $\Delta t = \sum_{i=1}^{h}(t_{max} - t_i) + \alpha \cdot \Delta h \cdot t_{payload}$
- **Bi-objective model**
  - $BIO = \Delta L + \beta \cdot t_{max} + \gamma \cdot \Delta t$
- **Cost of routing paths**
  - $\Delta L_3 < \Delta L_2 < \Delta L_1 < \Delta L_4$
  - $BIO_2 < BIO_3 < BIO_1 < BIO_4$



	k	Δh	h	t_{max}	Δt
P_1	1	0	3	4	5
P_2	2	0	3	2	1
P_3	2	2	5	5	8
P_4	3	6	9	5	31
Assume $t_{payload} = 5$ cycles, $\alpha = 0.5$					

Variant-2: Branch-and-bound bi-objective routing (3BOR)

- **Two-stage pipeline process**
- **Bounding function**
  - $BIO_x = BIO_{actual_i} + BIO_{actual_{i \to x}} + BIO_{min_{x \to D}}$
- **Pruning rules**
  - $BIO_x < Upper_{BIO}$
  - $\Delta L_x < \Delta L_{min} + max_{\Delta L}$
  - $\Delta t_x < h_s \cdot max_{\Delta t}$
- **Adjustable parameters**
  - Weight factor: α, β, γ
  - Pruning rules: $max_{\Delta L}, max_{\Delta t}$

Algorithm 2: Branch-and-Bound Bi-Objective Routing Optimization (3BOR)

input : global information TT and LT, routing request buffer RB and RB'

output: the bi-objective optimal path buffer PB

```
for Req' ∈ RB' do
    P_Bio ← Func3BOR(TT, LT, Req')
    PB.add(P_Bio)
for Req ∈ RB do
    P_ΔL ← FuncBBTR(TT, Req)
    Req' ← (Req, P_ΔL)
    RB'.add(Req')
```

Variant-2: Branch-and-bound bi-objective routing (3BOR)

- **Two-stage pipeline process**
- **Bounding function**
  - $BIO_x = BIO_{actual_i} + BIO_{actual_{i \to x}} + BIO_{min_{x \to D}}$
- **Pruning rules**
  - $BIO_x < \ Upper_{BIO}$
  - $\Delta L_x < \Delta L_{min} + max_{\Delta L}$
  - $\Delta t_x < h_s \cdot max_{\Delta t}$
- **Adjustable parameters**
  - Weight factor: α, β, γ
  - Pruning rules: $max_{\Delta L}, max_{\Delta t}$

Algorithm 2: Branch-and-Bound Bi-Objective Routing Optimization (3BOR)

input : global information TT and LT, routing request buffer RB and RB'
output: the bi-objective optimal path buffer PB

```
for Req′ ∈ RB′ do
    P_Bio ← Func3BOR(TT, LT, Req′)
    PB.add(P_Bio)
for Req ∈ RB do
    P_ΔL ← FuncBBTR(TT, Req)
    Req′ ← (Req, P_ΔL)
    RB′.add(Req′)
```

Variant-2: Branch-and-bound bi-objective routing (3BOR)

- **Two-stage pipeline process**
- **Bounding function**
  - $BIO_x = BIO_{actual_i} + BIO_{actual_{i \to x}} + BIO_{min_{x \to D}}$
- **Pruning rules**
  - $BIO_x < \ Upper_{BIO}$
  - $\Delta L_x < \Delta L_{min} + max_{\Delta L}$
  - $\Delta t_x < h_s \cdot max_{\Delta t}$
- **Adjustable parameters**
  - Weight factor: α, β, γ
  - Pruning rules: $max_{\Delta L}, max_{\Delta t}$

Algorithm 2: Branch-and-Bound Bi-Objective Routing Optimization (3BOR)

input : global information TT and LT, routing request buffer RB and RB'
output: the bi-objective optimal path buffer PB

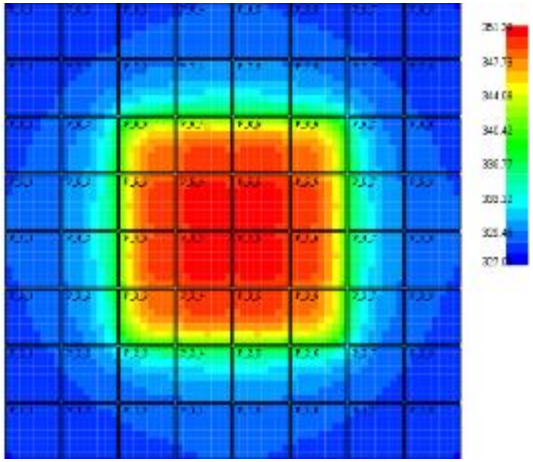
for $Req' \in RB'$ **do**
  $P_{Bio} \leftarrow$ `Func3BOR` (TT, LT, Req')
  $PB.add(P_{Bio})$
for $Req \in RB$ **do**
  $P_{\Delta L} \leftarrow$ `FuncBBTR` (TT, Req)
  $Req' \leftarrow (Req, P_{\Delta L})$
  $RB'.add(Req')$

Outline

- **Research background**
- **Branch and bound routing for ONoCs**
  - Branch-and-bound routing pruned at turning nodes
  - Branch-and-bound thermal-aware routing
  - Branch-and-bound bi-objective routing
- **Simulation results**
- **Conclusion and future works**

Simulation Experiment Configuration

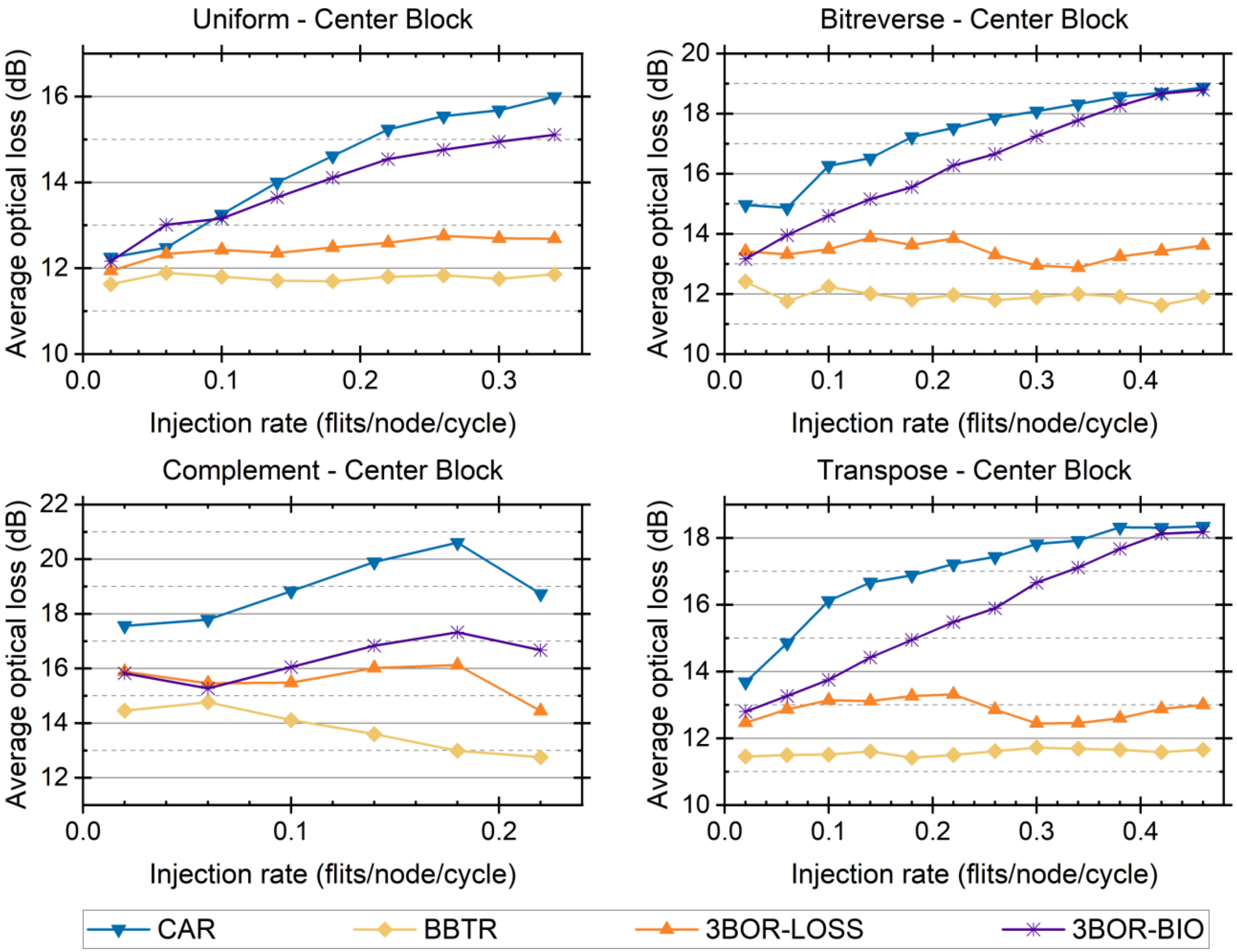
- **A cycle-accurate simulation platform based on SystemC**
- **Comparison baseline**
  - Contention-aware routing (CAR) [1]
- **Two variants of 3BOR through adjusting parameters**
  - 3BOR-LOSS
  - 3BOR-BIO
- **Temperature distribution**
  - Center-Block



[1] M. Li, W. Liu, L. H. Duong, P. Chen, L. Yang, and C. Xiao, “Contention-aware routing for thermal-reliable optical networks-on-chip,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 40, no. 2, pp. 260–273, 2020.

Simulation Experiment Configuration

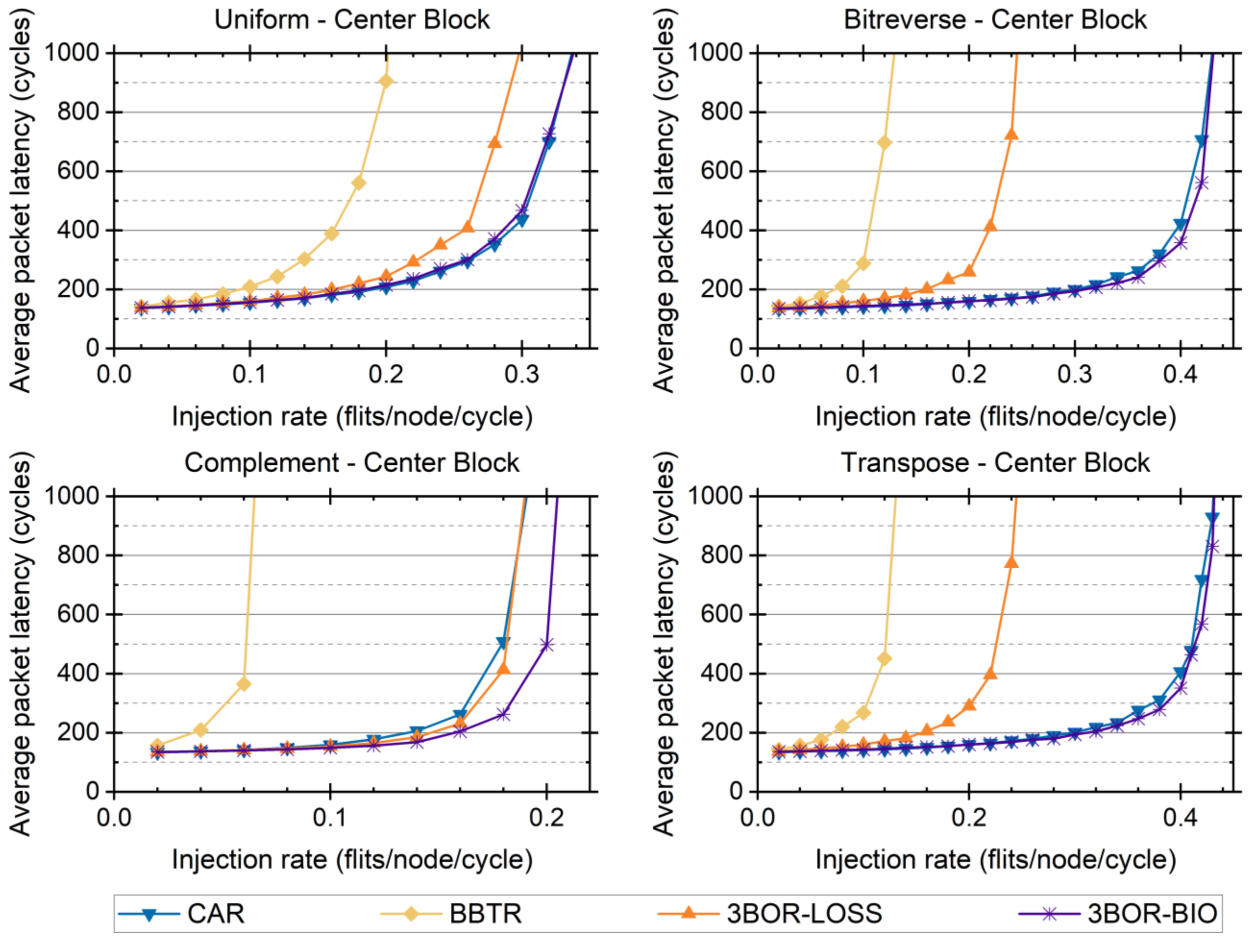
- **Average optical loss**
  - BBTR < 3BOR-LOSS < 3BOR-BIO < CAR



Average optical loss with varying injection rate

Simulation Experiment Configuration

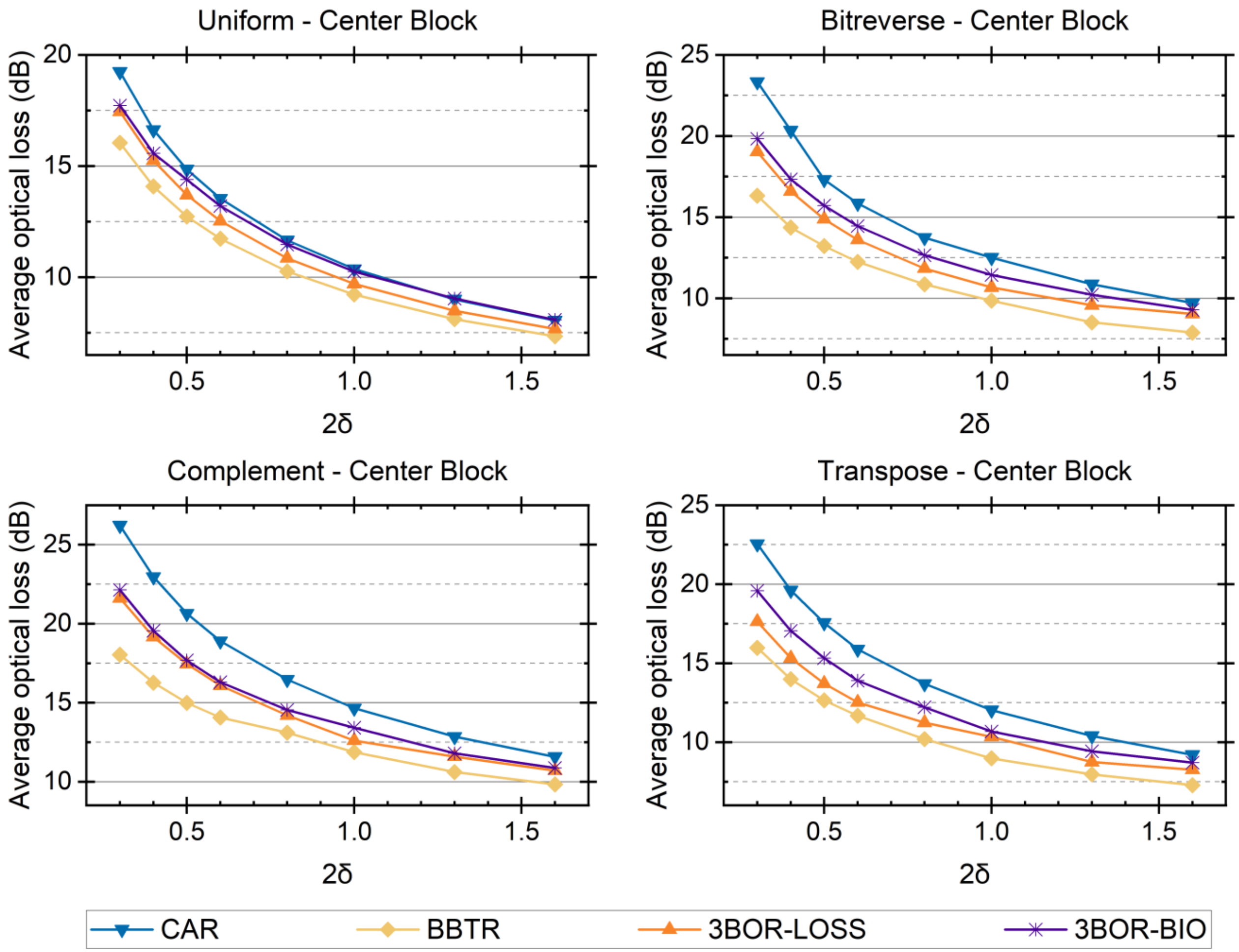
- **Average packet latency**
  - 3BOR-BIO < CAR < 3BOR-LOSS < BBTR



Average packet latency with varying injection rate

Simulation Experiment Configuration

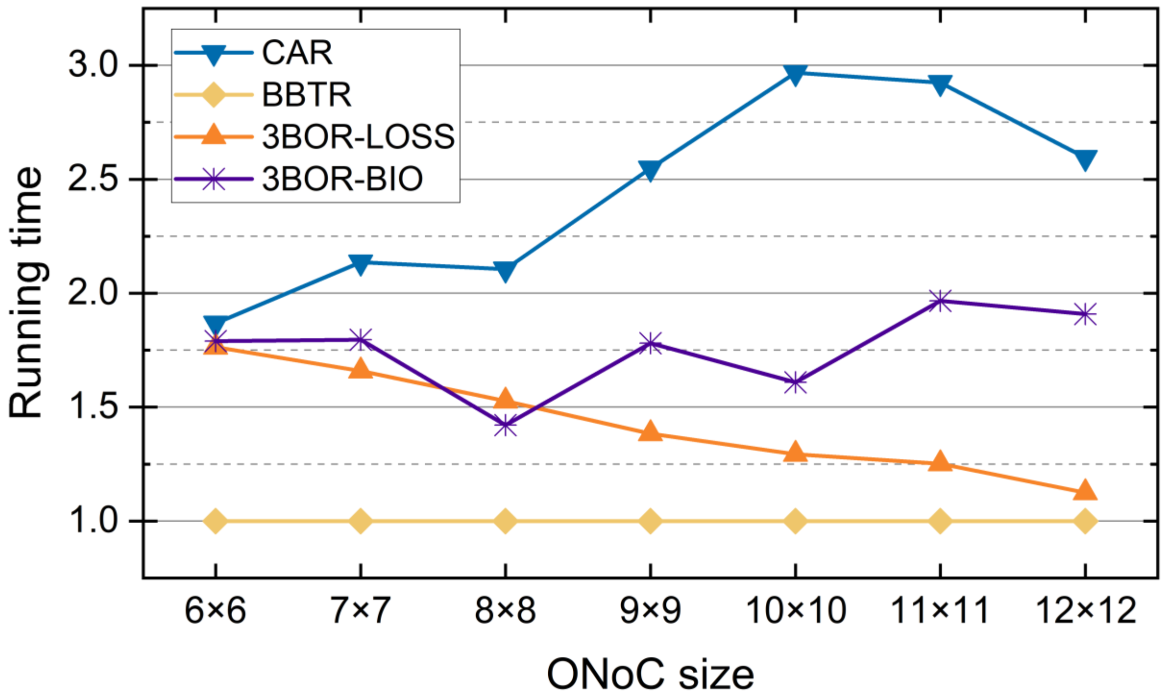
- **The proposed BBTR and 3BOR demonstrate universality in optimizing average optical loss**



Average optical loss with varying 3-dB bandwidth

Simulation Experiment Configuration

- **The proposed BBTR and 3BOR exhibits scalability**



Comparisons for algorithm running time on various ONoC sizes

Conclusions and Future works

- **Conclusion**
  - The proposed BBTR has the best optimization effect on average optical power loss
  - The proposed 3BOR can comprehensively optimize communication performance and energy efficiency
  - By adjusting parameters, 3BOR can balance the two optimization objectives, adapting to different network traffic patterns and chip temperature distributions.
- **Future works**
  - 3BOR with automatic parameter adjustment capability
  - New variants of BBR for optimizing different goals