

Chapter 7: Optimization (Routing and Wavelength Assignment)

Helmut Simonis

Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

ECLIPSe ELearning [Overview](#)



Licence

This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported License.

To view a copy of this license, visit [http:](http://creativecommons.org/licenses/by-nc-sa/3.0/)

[//creativecommons.org/licenses/by-nc-sa/3.0/](http://creativecommons.org/licenses/by-nc-sa/3.0/) or

send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.



Outline

- 1 Problem
- 2 Program
- 3 Search



What We Want to Introduce

- Optimization
- Graph algorithm library
- Problem decomposition
- Routing and Wavelength Assignment in Optical Networks



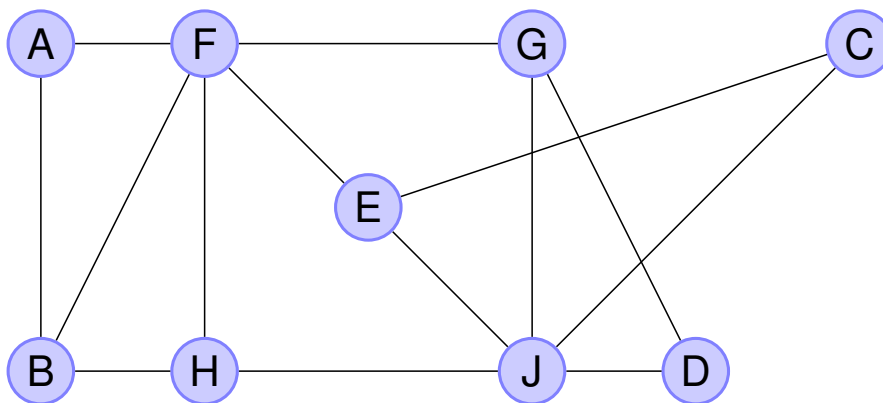
Problem Definition

Routing and Wavelength Assignment

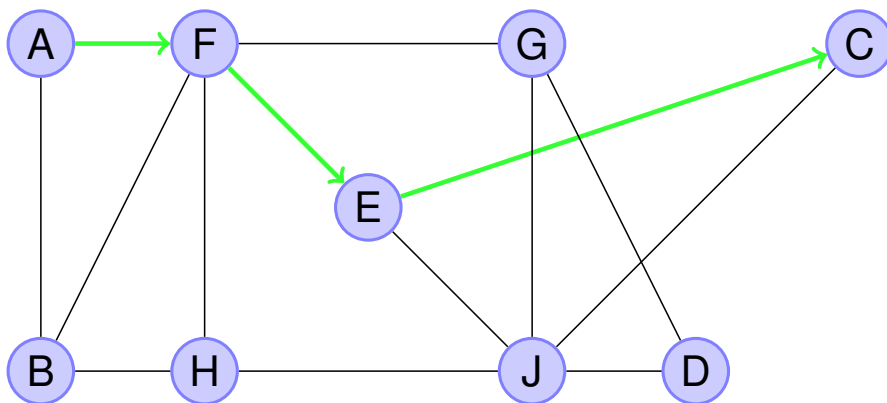
In an optical network, traffic demands between nodes are assigned to a route through the network and a specific wavelength. The route (called *lightpath*) must be a simple path from source to destination. Demands which are routed over the same link must be allocated to different wavelengths, but wavelengths may be reused for demands which do not meet. The objective is to find a combined routing and wavelength assignment which minimizes the number of wavelengths used for a given set of demands.



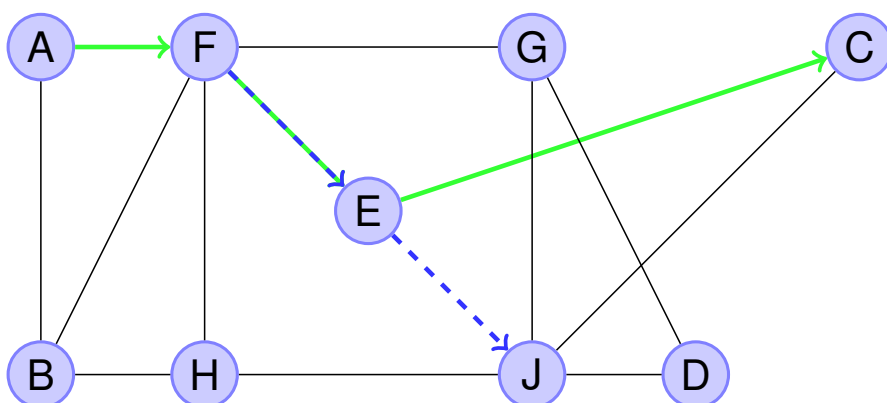
Example Network



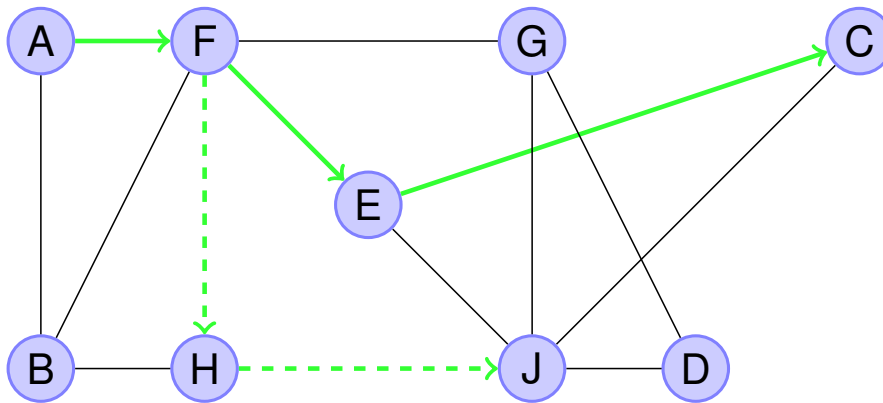
Lightpath from A to C



Conflict between demands A to C and F to J: Use different frequencies

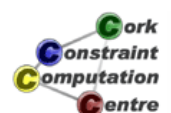


Conflict between demands A to C and F to J : Use different paths



Solution Approaches

- Greedy heuristic
- Optimization algorithm for complete problem
- **Decomposition into two problems**
 - Find routing
 - Assign wavelengths



Finding Routing

- Find routing which does not assign too many demands on the same link
- Lower bound for overall problem
- Do not use arbitrarily complex paths
- Start with shortest paths



Proposed Solution

- For each demand, use a shortest path between source and destination
- Shortest path = smallest number of links used
- Good for overall network utilisation
- May create bottlenecks on some links



How to Find Shortest Paths

- Well studied, well understood problem
- Many different algorithms for particular cases
 - Positive/negative weight
 - Path between pair of nodes/between node and all other nodes/between all nodes
 - One/all shortest paths or paths which are nearly shortest paths
- Don't program this yourself!
- Library in ECLiPSe: `lib(graph_algorithms)`



Library `graph_algorithms`

- Provides different algorithms about graphs
- Based on opaque `Graph` structure created from nodes and edges
- `make_graph(NrNodes, Edges, Graph)`
- Edges are terms `e(FromNode, ToNode, Weight)`
- Directed graphs as default, undirected graphs represented by edges in both directions



Basic Shortest Path Method

- `single_pair_shortest_path(Network, -1, From, To, Result)`
- Find path from node `From` to node `To` in graph `Network`
- Second argument describes weight function
 - -1: use number of hops
- `Result` given length of path and edges as list



Problem 2: Assign Wavelength

- Demands are routed on shortest paths
- Demands routed over the same link must have different frequencies
- Minimize maximal number of frequencies used



Model

- Domain variable for every demand
- Initial domain large, e.g. number of demands
- Disequality constraint between demands routed over same link
- Alternative: `alldifferent` constraints for all demands over each link
- Feasible solution: find assignment for variables



Optimization

- We are not looking for only a feasible solution
- We want to optimize objective
- Minimize largest value used



Library `branch_and_bound`

- `bb_min(Goal, Cost, bb_options{})`
- Goal **search goal**
 - Like `search/6` or `labeling/1` call
- Cost **objective (domain variable)**
- `bb_options` **optional parameters**
 - `timeout:Time` timeout limit in seconds
 - `from:LowerBound` known lower bound
 - `to:UpperBound` known upper bound



Example

```
...  
List :: 1..20,  
...  
ic:max(List,Max),  
bb_min(labeling(List),Max,  
        bb_options{timeout:100,from:10}),  
...
```



ic Constraint max (List, Var)

- Var is the largest value occurring in List
- Similar min (List, Var)
- Do not confuse with max in core language



Main Program

```
:-module (pure) .  
:-export (top/5) .  
:-lib (ic) .  
:-lib (ic_global) .  
:-lib (graph_algorithms) .  
:-lib (branch_and_bound) .
```

```
top (Name, NrDemands, LowerBound, Assignment, Max) :-  
    problem (Name, NrDemands, Network, Demands) ,  
    route (Network, Demands, Routes) ,  
    wave (NrDemands, Routes,  
          LowerBound, Assignment, Max) .
```



Routing

```
route(Network, Demands, Routes) :-  
    (foreach(demand(I, From, To), Demands),  
     foreach(route(I, Path), Routes),  
     param(Network) do  
         single_pair_shortest_path(Network, -1,  
                                     From, To,  
                                     _-Path)  
     ).
```



Wavelength Assignment

```
wave(NrDemands, Routes, LowerBound, Var, Max) :-  
    dim(Var, [NrDemands]),  
    Var[1..NrDemands] :: 1..NrDemands,  
    ic:max(Var, Max),  
    setup_alldifferent(Routes, Var, LowerBound),  
    bb_min(assign(Var), Max,  
            bb_options{from:LowerBound,  
                       timeout:100}).
```



Assignment Routine

```
assign(Var) :-  
    search(Var, 0, most_constrained, indomain,  
          complete, []).
```



Variable Selection Method `most_constrained`

- Similar to `first_fail`
- Select variable with smallest domain first
- For tie break, select variable in largest number of constraints



Creating alldifferent Constraints

```

setup_alldifferent (Routes, Var, LowerBound) :-
    (foreach (route (I, Path), Routes),
     fromto ([], A, A1, Pairs) do
        (foreach (Edge, Path),
         fromto (A, AA, [l (Edge, I) | AA], A1),
         param (I) do
             true
         )
        ),
    group (Pairs, l, Groups),
    ...

```

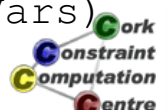


Creating alldifferent Constraints (II)

```

...
(foreach (_-Group, Groups),
 fromto (0, A, A1, LowerBound),
 param (Var) do
     length (Group, N),
     A1 is eclipse_language:max (N, A),
     (foreach (l (_, I), Group),
      foreach (X, AlldifferentVars),
      param (Var) do
          subscript (Var, [I], X)
      ),
     ic_global:alldifferent (AlldifferentVars)
).

```

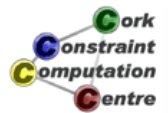


Generating Data

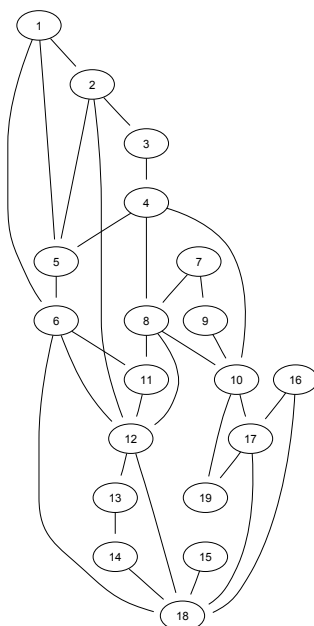
```

problem(Name, NrDemands, Network, Demands) :-
    network_topology(Name, NrNodes, Edges),
    make_graph(NrNodes, Edges, Directed),
    make_undirected_graph(Directed, Network),
    (for(I, 1, NrDemands),
     fromto([], A, [demand(I, From, To) | A], Demands),
     param(NrNodes) do
        repeat,
        From is 1+(random mod NrNodes),
        To is 1+(random mod NrNodes),
        From \= To,
        !
    ).

```



Example Network: MCI



MCI Topology Data

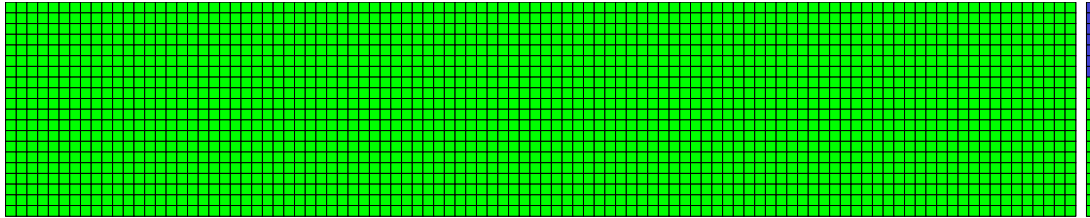
```
network_topology(mci, 19,
[e(1, 2, 1), e(1, 5, 1), e(1, 6, 1), e(2, 3, 1),
e(2, 5, 1), e(2, 12, 1), e(3, 4, 1), e(4, 5, 1),
e(4, 8, 1), e(4, 10, 1), e(5, 6, 1), e(6, 11, 1),
e(6, 12, 1), e(6, 18, 1), e(7, 8, 1), e(7, 9, 1),
e(8, 10, 1), e(8, 11, 1), e(8, 12, 1), e(9, 10, 1),
e(10, 17, 1), e(10, 19, 1), e(11, 12, 1), e(12, 13, 1),
e(12, 18, 1), e(13, 14, 1), e(14, 18, 1), e(15, 18, 1),
e(16, 17, 1), e(16, 18, 1), e(17, 18, 1), e(17, 19, 1)]) .
```



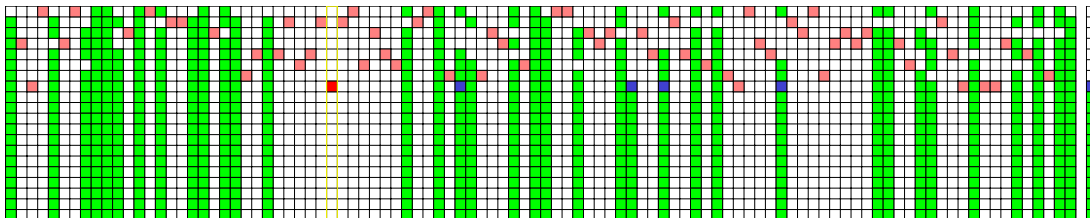
Searchtree



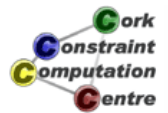
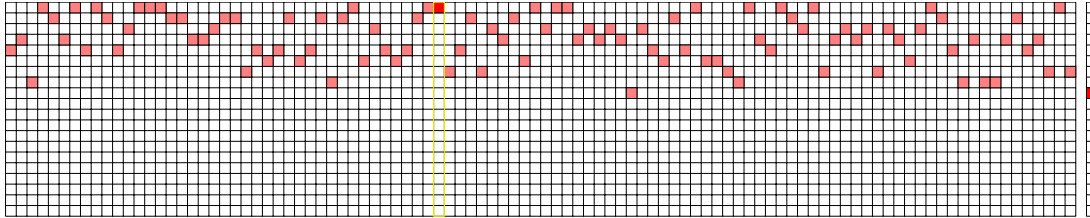
Initial State



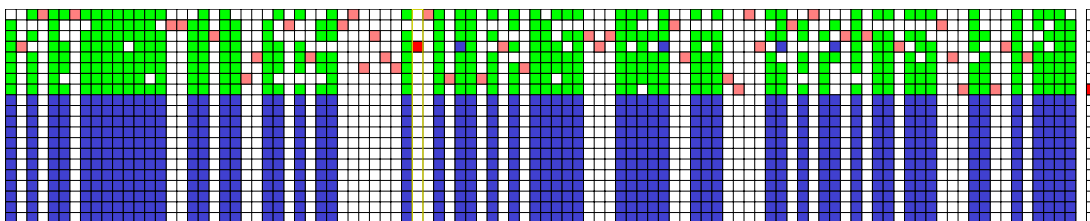
Update Cost



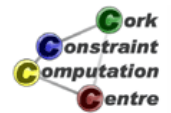
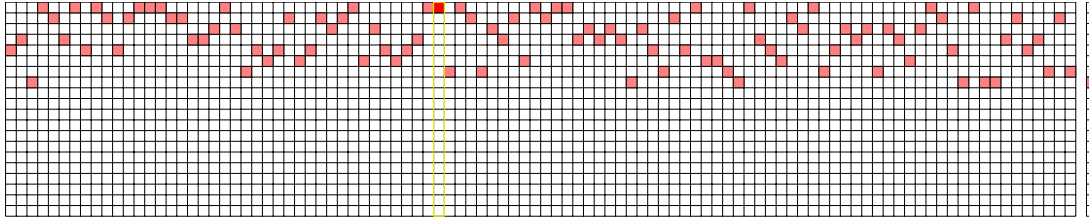
First Solution



Continue Search

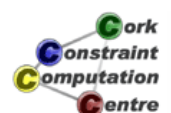


Optimal Solution



Observations

- Optimal solution found with minimal backtracking
- Reaching lower bound avoids enumeration proof of optimality
- Not guaranteed to be optimal for original problem
- Given decomposition destroys flexibility in finding solution



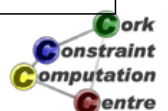
Further Experiments

- Vary number of demands to be handled
- Make 100 runs with randomized demands



Multiple Runs (100 experiments)

Network	Nr Demands	Avg LB	Avg Sol	σ Sol	Avg Gap
mci	20	3.71	3.71	0.711	0.00
mci	40	5.85	5.85	0.931	0.00
mci	60	7.69	7.69	1.324	0.00
mci	80	9.48	9.48	1.353	0.00
mci	100	11.34	11.34	1.687	0.00
mci	120	12.89	12.89	1.928	0.00
mci	140	14.59	14.59	2.298	0.00
mci	160	16.28	16.28	2.421	0.00
mci	180	17.89	17.89	2.656	0.00
mci	200	19.52	19.52	2.456	0.00



Observations

- These are not hard problem instances
- In general, graph coloring can be much more difficult
- Fast, simple solution to RWA problem
- Quality gap to be determined
 - Chapter 17: Solving RWA with MILP
 - Chapter 18: A Hybrid model for RWA



Network Problems

- `graph_algorithms` library
- Shortest path, articulation points, critical links
- Matching, strongly connected components
- Max-flow/min-cut
- Interface to AT&T graphviz visualizer





Optimization in ECLiPSe

- `branch_and_bound` library
- Not restricted to `ic` library
- Simple extension of search
- Importance of lower bounds
- For best results, needs support in constraint model





More Information

-  [Rajiv Ramaswami and Kumar N. Sivarajan.](#)
Routing and wavelength assignment in all-optical networks.
IEEE/ACM Trans. Netw., 3(5):489–500, 1995.
-  [Dhritiman Banerjee and Biswanath Mukherjee.](#)
A practical approach for routing and wavelength assignment in large wavelength-routed optical networks.
IEEE Journal on Selected Areas in Communications, 14(5):903–908, June 1996.





More Information

-  Brigitte Jaumard, Christophe Meyer, and Babacar Thiongane.
ILP formulations for the routing and wavelength assignment problem: Symmetric systems.
In M. Resende and P. Pardalos, editors, *Handbook of Optimization in Telecommunications*, pages 637–677. Springer, 2006.
-  Brigitte Jaumard, Christophe Meyer, and Babacar Thiongane.
Comparison of ILP formulations for the RWA problem.
Optical Switching and Networking, 4(3-4):157–172, 2007.



More Information

-  Helmut Simonis.
A hybrid constraint model for the routing and wavelength assignment problem.
CP 2009, Lisbon, September 2009.
<http://4c.ucc.ie/~hsimonis/rwa.pdf>
-  Helmut Simonis.
Solving the static design routing and wavelength assignment problem.
CSCLP 2009, Barcelona, Spain, June 2009.

