Chapter 7: Optimization (Routing and Wavelength Assignment)

Helmut Simonis

Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

ECLiPSe ELEarning

Overview
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://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
Outline

1. Problem
   - Problem 1: Find routing
   - Problem 2: Assign Wavelengths

2. Program

3. Search
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

Problem 1: Find routing
Problem 2: Assign Wavelengths

Problem
Program
Search
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
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 bottlelenecks 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
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 \texttt{branch\_and\_bound}

- \texttt{bb\_min} (\texttt{Goal, Cost, bb\_options{}})
- \texttt{Goal} \texttt{search goal}
  - \texttt{Like} \texttt{search/6} or \texttt{labeling/1} call
- \texttt{Cost} \texttt{objective (domain variable)}
- \texttt{bb\_options} \texttt{optional parameters}
  - \texttt{timeout:Time} timeout limit in seconds
  - \texttt{from:LowerBound} known lower bound
  - \texttt{to:UpperBound} known upper bound
Example

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

Helmut Simonis  Optimization
\textbf{ic Constraint }\texttt{max(List,Var)}

- \texttt{Var} is the largest value occurring in \texttt{List}
- \texttt{Similar} \texttt{min(List,Var)}
- Do not confuse with \texttt{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).
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)
    ).
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}).
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

```prolog
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, 1, Groups),
  ...
```
... 

(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

```
probem(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
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)])

Outline

1. Problem
2. Program
3. Search
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree

Cost Update
First Solution
Skip Animation

Back to Start

Helmut Simonis
Optimization
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
The image displays a search tree structure with nodes and edges. The nodes are represented by green squares with numbers inside, and the edges connect these nodes. The tree appears to be traversed in a certain order, indicated by the highlighted nodes.

The table below represents a portion of the tree:

| 4 | 16 | 55 | 94 | 96 | 17 | 20 | 47 | 49 | 76 | 89 | 32 | 35 | 71 | 36 | 23 | 69 | 7 | 2 | 74 | 79 | 24 | 34 | 68 | 93 | 70 | 63 | 81 | 42 | 40 | 56 | 57 | 27 | 64 | 33 | 84 | 88 | 29 | 37 | 45 | 90 | 39 |

### Search Tree

1. Start at the root node.
2. Move down the tree, following the edges.
3. Highlight the nodes as you traverse.
4. Continue until you reach the desired node or solution.

---

Helmut Simonis  Optimization

Page 78
Searchtree
Searchtree
Searchtree
Searchtree

Back to Start  Cost Update  First Solution  Skip Animation
Searchtree
Searchtree
| 4 | 16 | 55 | 94 | 96 | 17 | 20 | 47 | 49 | 76 | 89 | 32 | 35 | 71 | 36 | 23 | 69 | 7 | 2 | 74 | 79 | 24 | 34 | 68 | 93 | 70 | 63 | 81 | 42 | 40 | 56 | 57 | 27 | 64 | 33 | 84 | 88 | 29 | 37 | 45 | 90 | 39 | 78 | 52 | 6 | 26 | 28 | 30 | 75 | 14 | 80 | 60 | 61 | 66 | 77 | 7 | 6 | 5 | 2 | 4 | 3 | 2 | 6 | 5 | 3 | 2 | 4 | 1 | 5 | 2 | 3 | 4 | 1 | 7 | 3 | 2 | 1 | 8 | 7 | 6 | 5 | 3 | 2 | 4 | 1 | 8 | 7 | 5 | 4 | 3 | 2 | 5 | 1 | 6 | 4 | 3 | 2 | 5 | 4 | 3 | 2 | 1 |
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree

---

Back to Start  Cost Update  First Solution  Skip Animation
Searchtree
Searchtree
| 4 | 16 | 55 | 94 | 96 | 17 | 20 | 47 | 49 | 76 | 89 | 32 | 35 | 71 | 36 | 23 | 69 | 7 | 2 | 74 | 79 | 24 | 34 | 68 | 93 | 70 | 63 | 81 | 42 | 40 | 56 | 57 | 27 | 64 | 33 | 84 | 88 | 29 | 37 | 45 | 90 | 39 | 78 | 52 | 6 | 26 | 28 | 30 | 75 | 14 | 80 | 60 | 61 | 66 | 77 | 92 | 46 | 53 | 72 | 85 | 98 | 3 | 12 | 31 | 59 | 73 | 43 | 62 | 44 | 11 | 86 | 54 | 8 | 87 | 91 | 65 | 2 | 9 | 8 | 7 | 6 | 5 | 1 | 3 | 8 | 7 | 6 | 5 | 3 | 4 | 1 | 3 | 2 | 6 | 5 | 4 | 1 | 4 | 2 | 8 | 7 | 6 | 5 | 2 | 4 | 1 | 5 | 2 | 3 | 4 | 1 | 7 | 3 | 2 | 1 | 8 | 7 | 6 | 5 | 3 | 2 | 4 | 1 | 8 | 7 | 5 | 4 | 3 | 2 | 5 | 1 | 6 | 4 | 3 | 2 | 5 | 4 | 3 | 2 | 1 | 8 | 7 | 6 | 5 | 3 | 2 | 4 | 1 | 8 | 7 | 5 | 4 | 3 | 2 | 5 | 1 | 6 | 4 | 3 | 2 | 5 | 4 | 3 | 2 | 1 |

Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
| 4   | 16 | 55 | 94 | 96 | 17 | 20 | 47 | 49 | 76 | 89 | 32 | 35 | 71 | 36 | 23 | 69 | 7 | 2 | 74 | 79 | 24 | 34 | 68 | 93 | 70 | 63 | 81 | 42 | 40 | 56 | 57 | 27 | 64 | 33 | 84 | 88 | 29 | 37 | 45 | 90 | 39 | 78 | 52 | 6 | 26 | 28 | 30 | 75 | 14 | 80 | 60 | 61 | 66 | 77 | 92 | 46 | 53 | 72 | 85 | 98 | 3 | 12 | 31 | 59 | 73 | 43 | 62 | 44 | 11 | 86 | 54 | 8 | 87 | 91 | 65 | 83 | 38 | 67 | 82 | 15 | 99 | 48 | 95 | 97 | 22 | 51 | 19 | 18 | 1 | 25 | 100 | 5 | 21 | 58 | 50 | 9 | 10 | 13 | 41 | 1 | 2 | 1 | 2 | 1 | 4 | 2 | 2 | 7 | 6 | 5 | 4 | 4 | 2 | 2 | 1 | 1 | 7 | 6 | 5 | 3 | 1 | 4 | 1 | 5 | 4 | 3 | 5 | 2 | 6 | 5 | 1 | 9 | 8 | 7 | 6 | 5 | 1 | 3 | 8 | 7 | 6 | 5 | 3 | 2 | 4 | 1 | 8 | 7 | 5 | 4 | 3 | 2 | 5 | 4 | 3 | 2 | 1 | 8 | 7 | 6 | 5 | 3 | 2 | 4 | 1 | 8 | 7 | 5 | 4 | 3 | 2 | 5 | 4 | 3 | 2 | 1 |

The diagram represents a search tree, likely used to illustrate the process of searching for a solution to a problem. The numbers correspond to nodes in the tree, and the structure demonstrates the branching and decision-making process.
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
null
Searchtree
Searchtree
Searchtree
Searchtree
<table>
<thead>
<tr>
<th>15</th>
<th>26</th>
<th>37</th>
<th>48</th>
<th>59</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>27</td>
<td>38</td>
<td>49</td>
<td>60</td>
<td>71</td>
</tr>
<tr>
<td>55</td>
<td>66</td>
<td>77</td>
<td>88</td>
<td>99</td>
<td>100</td>
</tr>
<tr>
<td>94</td>
<td>76</td>
<td>89</td>
<td>32</td>
<td>35</td>
<td>36</td>
</tr>
<tr>
<td>96</td>
<td>47</td>
<td>49</td>
<td>76</td>
<td>89</td>
<td>94</td>
</tr>
<tr>
<td>17</td>
<td>20</td>
<td>47</td>
<td>49</td>
<td>76</td>
<td>89</td>
</tr>
<tr>
<td>23</td>
<td>69</td>
<td>71</td>
<td>32</td>
<td>35</td>
<td>36</td>
</tr>
<tr>
<td>69</td>
<td>71</td>
<td>32</td>
<td>35</td>
<td>36</td>
<td>23</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>74</td>
<td>79</td>
<td>24</td>
<td>34</td>
</tr>
<tr>
<td>74</td>
<td>79</td>
<td>24</td>
<td>34</td>
<td>68</td>
<td>93</td>
</tr>
<tr>
<td>79</td>
<td>24</td>
<td>34</td>
<td>68</td>
<td>93</td>
<td>74</td>
</tr>
<tr>
<td>24</td>
<td>34</td>
<td>68</td>
<td>93</td>
<td>74</td>
<td>79</td>
</tr>
<tr>
<td>34</td>
<td>68</td>
<td>93</td>
<td>74</td>
<td>79</td>
<td>24</td>
</tr>
<tr>
<td>68</td>
<td>93</td>
<td>74</td>
<td>79</td>
<td>24</td>
<td>34</td>
</tr>
<tr>
<td>93</td>
<td>74</td>
<td>79</td>
<td>24</td>
<td>34</td>
<td>68</td>
</tr>
<tr>
<td>74</td>
<td>79</td>
<td>24</td>
<td>34</td>
<td>68</td>
<td>93</td>
</tr>
<tr>
<td>79</td>
<td>24</td>
<td>34</td>
<td>68</td>
<td>93</td>
<td>74</td>
</tr>
<tr>
<td>24</td>
<td>34</td>
<td>68</td>
<td>93</td>
<td>74</td>
<td>79</td>
</tr>
<tr>
<td>34</td>
<td>68</td>
<td>93</td>
<td>74</td>
<td>79</td>
<td>24</td>
</tr>
<tr>
<td>68</td>
<td>93</td>
<td>74</td>
<td>79</td>
<td>24</td>
<td>34</td>
</tr>
<tr>
<td>93</td>
<td>74</td>
<td>79</td>
<td>24</td>
<td>34</td>
<td>68</td>
</tr>
<tr>
<td>74</td>
<td>79</td>
<td>24</td>
<td>34</td>
<td>68</td>
<td>93</td>
</tr>
<tr>
<td>79</td>
<td>24</td>
<td>34</td>
<td>68</td>
<td>93</td>
<td>74</td>
</tr>
<tr>
<td>24</td>
<td>34</td>
<td>68</td>
<td>93</td>
<td>74</td>
<td>79</td>
</tr>
<tr>
<td>34</td>
<td>68</td>
<td>93</td>
<td>74</td>
<td>79</td>
<td>24</td>
</tr>
<tr>
<td>68</td>
<td>93</td>
<td>74</td>
<td>79</td>
<td>24</td>
<td>34</td>
</tr>
<tr>
<td>93</td>
<td>74</td>
<td>79</td>
<td>24</td>
<td>34</td>
<td>68</td>
</tr>
<tr>
<td>74</td>
<td>79</td>
<td>24</td>
<td>34</td>
<td>68</td>
<td>93</td>
</tr>
<tr>
<td>79</td>
<td>24</td>
<td>34</td>
<td>68</td>
<td>93</td>
<td>74</td>
</tr>
<tr>
<td>24</td>
<td>34</td>
<td>68</td>
<td>93</td>
<td>74</td>
<td>79</td>
</tr>
<tr>
<td>34</td>
<td>68</td>
<td>93</td>
<td>74</td>
<td>79</td>
<td>24</td>
</tr>
<tr>
<td>68</td>
<td>93</td>
<td>74</td>
<td>79</td>
<td>24</td>
<td>34</td>
</tr>
<tr>
<td>93</td>
<td>74</td>
<td>79</td>
<td>24</td>
<td>34</td>
<td>68</td>
</tr>
<tr>
<td>74</td>
<td>79</td>
<td>24</td>
<td>34</td>
<td>68</td>
<td>93</td>
</tr>
<tr>
<td>79</td>
<td>24</td>
<td>34</td>
<td>68</td>
<td>93</td>
<td>74</td>
</tr>
<tr>
<td>24</td>
<td>34</td>
<td>68</td>
<td>93</td>
<td>74</td>
<td>79</td>
</tr>
<tr>
<td>34</td>
<td>68</td>
<td>93</td>
<td>74</td>
<td>79</td>
<td>24</td>
</tr>
<tr>
<td>68</td>
<td>93</td>
<td>74</td>
<td>79</td>
<td>24</td>
<td>34</td>
</tr>
</tbody>
</table>
Searchtree

[Diagram of a search tree with nodes and branches]

Back to Start  Cost Update  First Solution

Helmut Simonis  Optimization
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)

<table>
<thead>
<tr>
<th>Network</th>
<th>Nr Demands</th>
<th>Avg LB</th>
<th>Avg Sol</th>
<th>Σ Sol</th>
<th>Avg Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>mci</td>
<td>20</td>
<td>3.71</td>
<td>3.71</td>
<td>0.711</td>
<td>0.00</td>
</tr>
<tr>
<td>mci</td>
<td>40</td>
<td>5.85</td>
<td>5.85</td>
<td>0.931</td>
<td>0.00</td>
</tr>
<tr>
<td>mci</td>
<td>60</td>
<td>7.69</td>
<td>7.69</td>
<td>1.324</td>
<td>0.00</td>
</tr>
<tr>
<td>mci</td>
<td>80</td>
<td>9.48</td>
<td>9.48</td>
<td>1.353</td>
<td>0.00</td>
</tr>
<tr>
<td>mci</td>
<td>100</td>
<td>11.34</td>
<td>11.34</td>
<td>1.687</td>
<td>0.00</td>
</tr>
<tr>
<td>mci</td>
<td>120</td>
<td>12.89</td>
<td>12.89</td>
<td>1.928</td>
<td>0.00</td>
</tr>
<tr>
<td>mci</td>
<td>140</td>
<td>14.59</td>
<td>14.59</td>
<td>2.298</td>
<td>0.00</td>
</tr>
<tr>
<td>mci</td>
<td>160</td>
<td>16.28</td>
<td>16.28</td>
<td>2.421</td>
<td>0.00</td>
</tr>
<tr>
<td>mci</td>
<td>180</td>
<td>17.89</td>
<td>17.89</td>
<td>2.656</td>
<td>0.00</td>
</tr>
<tr>
<td>mci</td>
<td>200</td>
<td>19.52</td>
<td>19.52</td>
<td>2.456</td>
<td>0.00</td>
</tr>
</tbody>
</table>
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