Chapter 7: Optimization (Routing and Wavelength Assignment)

Helmut Simonis

Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

ECLiPSe ELearning
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
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

A - F - G - C
B - H - J - D
E
Lightpath from A to C

Problem 1: Find routing
Problem 2: Assign Wavelengths
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 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: \textit{alldifferent} constraints for all demands over each link
- Feasible solution: find assignment for variables
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 $\text{max}(\text{List}, \text{Var})$

- $\text{Var}$ is the largest value occurring in $\text{List}$
- Similar $\text{min}(\text{List}, \text{Var})$
- Do not confuse with $\text{max}$ in core language
Outline

1. Problem
2. Program
3. Search
:-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),
    ...
```

Helmut Simonis
Optimization
...  
(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)
).
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
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
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree

Back to Start  Cost Update  First Solution  Skip Animation

Helmut Simonis  Optimization
Searchtree
Searchtree
Searchtree

---

[Diagram of a search tree with nodes and edges representing the search process.]
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
<table>
<thead>
<tr>
<th>4</th>
<th>16</th>
<th>55</th>
<th>94</th>
<th>96</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>20</td>
<td>47</td>
<td>49</td>
<td>76</td>
</tr>
<tr>
<td>89</td>
<td>32</td>
<td>35</td>
<td>71</td>
<td>36</td>
</tr>
<tr>
<td>23</td>
<td>69</td>
<td>7</td>
<td>2</td>
<td>74</td>
</tr>
<tr>
<td>79</td>
<td>24</td>
<td>34</td>
<td>68</td>
<td>93</td>
</tr>
<tr>
<td>70</td>
<td>63</td>
<td>81</td>
<td>42</td>
<td>40</td>
</tr>
<tr>
<td>56</td>
<td>57</td>
<td>27</td>
<td>64</td>
<td>33</td>
</tr>
<tr>
<td>84</td>
<td>88</td>
<td>29</td>
<td>37</td>
<td>45</td>
</tr>
<tr>
<td>90</td>
<td>39</td>
<td>78</td>
<td>52</td>
<td>6</td>
</tr>
<tr>
<td>26</td>
<td>28</td>
<td>30</td>
<td>75</td>
<td>14</td>
</tr>
<tr>
<td>80</td>
<td>60</td>
<td>61</td>
<td>66</td>
<td>77</td>
</tr>
<tr>
<td>92</td>
<td>46</td>
<td>53</td>
<td>72</td>
<td>85</td>
</tr>
<tr>
<td>98</td>
<td>3</td>
<td>12</td>
<td>31</td>
<td>59</td>
</tr>
<tr>
<td>73</td>
<td>43</td>
<td>62</td>
<td>44</td>
<td>11</td>
</tr>
<tr>
<td>86</td>
<td>54</td>
<td>8</td>
<td>87</td>
<td>91</td>
</tr>
<tr>
<td>65</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>7</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>7</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
### Searchtree

| 1 | 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 | 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 |

![Searchtree Diagram](image-url)

- Back to Start
- Cost Update
- First Solution
- Skip Animation

Helmut Simonis

Optimization
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
Searchtree
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
  Chapter 17: Solving RWA with MILP
  Chapter 18: A Hybrid model for RWA
Conclusions
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
Conclusions

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

Conclusions

More Information

Brigitte Jaumard, Christophe Meyer, and Babacar Thiongane.
ILP formulations for the routing and wavelength assignment problem: Symmetric systems.

Brigitte Jaumard, Christophe Meyer, and Babacar Thiongane.
Helmut Simonis.
A hybrid constraint model for the routing and wavelength assignment problem.
http://4c.ucc.ie/~hsimonis/rwa.pdf

Helmut Simonis.
Solving the static design routing and wavelength assignment problem.
CSCLP 2009, Barcelona, Spain, June 2009.