

Constraint Applications in Networks

Helmut Simonis

CrossCore Optimization Ltd

London

UK

email:helmut.simonis@crosscoreop.com

In this chapter we discuss the use of Constraint Programming (CP) for network applications. Network problems arise in many different domains, we take a rather narrow view in this presentation and concentrate on three areas:

- electrical networks
- water (oil) networks
- data networks

Some of the earliest examples for CLP(R) were for analysing analog circuits, whose behavior can be described by *Ohm's law* for the relation between resistance, voltage and current and *Kirchhoff's laws* which defines how connected circuits behave. This early analysis later was extended to more complex, hybrid networks. One of the most interesting practical problems in this domain is the configuration of electrical power distribution networks (section 1), which has been studied with different constraint techniques and for which operational systems exist.

The work on CP models for electrical power distribution networks later led to the study of water distribution networks (section 2), which have similar importance in the utilities area. Water networks differ from electrical networks in a number of fundamental ways:

- Water can be stored in the network, reservoirs and water towers are key elements of water distribution systems.
- Water quality can vary. Keeping track of relevant properties is important when mixing water from different sources.
- Many water networks were built in the 19th century. Old pipes may be leaking, losing up to 30 % of the water sent through them. The exact loss rate is typically unknown.

Networks for oil distribution have similar properties, but face additional challenges.

The bulk of constraint applications for networks Simonis [2004] are in the context of data networks (section 3), covering either traditional, connection oriented networks or packet-switched, routed networks like the Internet. We look at a number of different applications in this domain:

- We start with a problem of *application placement* (section 3.1) for the Italian Inter banking network, which was one of the first large scale CLP applications in the network domain.

- In many networks, the task of *path placement* (section 3.2) is to define the route on which a demand will be sent through the network. This is a fundamental networking problem, for which many competing CP methods have been proposed.
- One possible extension is the use of *multiple paths* (section 3.3) for demands, where the secondary path is only active when the primary connection has failed.
- Another possible extension is to add a time dimension, where traffic demands have given start and end times, and demands compete for network bandwidth if they overlap in time. This application is called *Bandwidth on Demand* (section 3.4).
- In the previous problems, the network structure and capacity was fixed. The problem of *Network Design* (section 3.5) deals with defining connectivity and finding the right link capacity to satisfy a projected set of demands.
- IP (Internet Protocol) networks usually do not use explicit routes for traffic demands. Instead, packets are routed based on a distributed shortest path algorithm. *Metric optimization* (section 3.6) deals with choosing metric weights to influence the routing in the network and to optimize the network utilization.
- Many traditional algorithms assume a demand matrix of all communication needs between network nodes. In IP networks this demand matrix is not readily available, *Resilience Analysis* (section 3.7) tries to predict network behavior in failure scenarios without explicit knowledge of the demand structure.
- Secondary paths and routing algorithms provide some methods to maintain network communications in case of element failure. The idea of *Bandwidth Protection* (section 3.8) offers an alternative, purely local mechanism for improving network resilience.

1 Electricity Networks

Electrical networks are controlled by the application of three fundamental relations, Ohm's law and the two Kirchhoff's laws:

Ohm's Law Ohm's Law relates the three fundamental electrical quantities: voltage V , current I and resistance Ω , $V = I * \Omega$.

Kirchhoff's Current Law The current flowing into a node or branching point is equal to the sum of the currents leaving the node or branching point.

Kirchhoff's Voltage Law The sum of all the voltages around any closed path in a circuit equals zero.

As an early example for CLP(R), it was shown in Heintze et al. [1987] (see also Jaffar and Maher [1994], Marriott and Stuckey [1998]), that simple electrical networks can be modeled using linear constraints over continuous domains. The use of constraints allows a very flexible query structure, were given, partial information about voltages, currents and resistance can be used to deduce the missing

information. The solver in CLP(R) was restricted to linear constraints only, so that most interesting electrical and electronic circuits can not be modeled. By combining the finite domain and the continuous solver in CHIP, the LOGICIM tool Graf et al. [1989] extended the modeling capabilities to hybrid circuits with state. It is perhaps surprising that after the development of non-linear solvers in the nineties (see chapter 16 “Continuous and Interval Constraints”) there has not been a systematic study of the simulation of analog electronic systems with Constraint Programming.

On the other hand there are application problems where the basic laws above are enough to model quite complex electrical circuits. This is the case for electrical power distribution networks, first studied for CLP in Creemers et al. [1995, 1998]. Power distribution networks are formed of high (medium) voltage transmission lines, which link power stations (the producers) to transformer stations (the consumers), which convert the high voltage to lower voltage for actual consumers. The transmission lines form a graph, of which only a subgraph is in active use at any one time. Lines can be enabled and disabled by switches, which can be opened or closed as required. At any time point, the active transmission network forms a forest, with trees rooted in the power stations and consumers at the leaves of the network. Cycles in the network must be avoided, since they will result in malfunctions and equipment damage. The configuration task for a power network consists in controlling the transmission switches in such a way that

- there are no cycles
- all customers are reached
- the capacity of the power stations is not exceeded
- the currents limits of the transmission lines are not exceeded
- loss due to resistance in the transmission lines is minimized

If an element failure occurs in a network, then there is a reconfiguration task to restore supply to all or at least to the most important customers as quickly as possible, while minimizing the number of switch changes. Reaction times in seconds (or faster) is required as a rule.

For maintenance or extension work it is necessary to isolate certain components in the network, so that they are safe to work on. Given a set of such maintenance jobs, we have a planning problem to define the best schedule of operations. This is typically further constrained by release and due dates, resource and manpower limits and other scheduling constraints.

The PlaNets system Creemers et al. [1995, 1998] was developed by the University of Catalonia in Barcelona (UCB) for the Spanish electricity company Enher to tackle these problems. It uses the rational solver of CHIP Dincbas et al. [1988] for the electrical network constraints, and the finite domain solver for the temporal and scheduling aspects of the system.

Recently, other constraints approaches for the reconfiguration problems have been attempted. In Hadzic and Andersen [2005], the use of a Boolean, BDD Bryant [1986] based solver is advocated to allow fast reaction times. The use of propositional calculus to represent open or closed switches is natural, but in

order to be able to use the Boolean solver also for the electrical constraints, massive simplifications are required. For example, all consumers are assigned the same, unit demand size, and Kirchoff's current laws are represented with small integers. The system currently is in experimental use at the Danish power company NESAs.

2 Water(Oil) Networks

The team at UCB working on the PlaNets system cooperated with experts on water distribution systems to form the CLOCWISe consortium to study the use of constraint programming for operational control of water systems Brdys et al. [2003]. Water distribution systems share many aspects with electrical power supply networks. The system forms a network with supply stations (water wells), transmission lines (pipes) and distribution nodes. The flow through nodes follows flow conservation (Kirchoff's current law), there are capacity limits for wells and pipes, etc. The scheduling scenarios are similar as well, there are configuration and reconfiguration tasks, and more complex planning scenarios. But there are also significant differences:

- Water supply systems use storage facilities, reservoirs and water towers. Levels in these storage nodes evolve over time, producer/consumer models Simonis and Cornelissens [1995] can be used to describe these constraints.
- Pipes can be operated at different flow rates, with significant differences in the energy expended.
- Many pipes are old and therefore leaky, losing water at an unknown, but significant rate. Therefore flow conservation is not preserved, the flow of water entering a pipe is different from the flow leaving the pipe.
- Chemical properties of water from different sources vary, and may be further changed by processing in the network. Mixing laws are quite complex and often non-linear.

The CLOCWISe system deals with these additional constraints using the rational solver of CHIP and its finite domain solver, especially global constraints for scheduling. A quality estimator for water mixing scenarios uses the linear solver, dealing with non-linear constraints by piece-wise linear approximation.

Pipeline transport for petro-chemical products is closely related. The FORWARD C system of Technip Simonis [1999] contains a module for pipeline transport between a tank farm at a deep water harbor and the tank farm associated with the refinery. The transmission plan can be seen as a sequencing operation in a schedule. Crude oil mixing plays an important role, as it can be used to optimize the throughput of the refinery by mixing crudes with different properties to more closely match the design specification of the refinery.

Studies to schedule more general oil distribution networks with Constraint Programming were largely unsuccessful. There are many non-linear, continuous constraints, the pipes can be used bi-directionally, requiring a detailed view of

the contents of the pipe at any given time. When sending two different products, one after the other, through the pipe, the products partially mix at the interface, requiring down-grading of the product, or even producing scrap that needs to be reworked at a refinery. An exception was the pipeline scheduling tool developed by PrologIA for AirLiquide, which was designed to handle parts of their European pipeline system.

3 Data Networks

We now consider data network applications, which form the core of this chapter.

3.1 Application Placement

The system described in Chiopris and Fabris [1994] was one of the first constraint applications dealing with networks. It was developed for the Italian Inter-banking network and deals with the placement of applications on servers in the network.

The problem is to place a set of applications \mathbf{K} on different servers in a network, so that they can run over-night batch jobs with data transmitted by users on nodes in the network. Each server j has limited CPU capacity, denoted $\text{cpu}(j)$, while many nodes of the network can not host any applications. Each user request states that a user on node i needs $\text{dem}(i, k)$ CPU units for application k . If we run application k on a server j , we have to pay a cost $\text{lic}(j, k)$, typically license fees. Transmitting data from user i to server j creates a transport cost $\text{dist}(i, j)$ per unit transported.

We introduce $\{0, 1\}$ decision variables $A_{j,k}$ to indicate if application k is running on server j , and decision variables U_{ij}^k to indicate that the demand of user i for application k is satisfied by server j . We can then express the model in this form

$$\min_{\{A_{jk}, U_{ij}^k\}} \sum_{k \in \mathbf{K}} \sum_{j \in \mathbf{N}} \text{lic}(j, k) A_{jk} + \sum_{k \in \mathbf{K}} \sum_{i \in \mathbf{N}} \sum_{j \in \mathbf{N}} \text{dist}(i, j) \text{dem}(i, k) U_{ij}^k \quad (1)$$

st.

$$\forall j \in \mathbf{N} : \sum_{k \in \mathbf{K}} \sum_{i \in \mathbf{N}} \text{dem}(i, k) U_{ij}^k \leq \text{cpu}(j) \quad (2)$$

$$\forall k \in \mathbf{K}, \forall i \in \mathbf{N} : \sum_{j \in \mathbf{N}} U_{ij}^k = 1 \quad (3)$$

$$\forall k \in \mathbf{K}, \forall i \in \mathbf{N}, \forall j \in \mathbf{N} : U_{ij}^k \implies A_{jk} \quad (4)$$

$$A_{jk} \in \{0, 1\}$$

$$U_{ij}^k \in \{0, 1\}$$

The objective function (1) is to minimize the total cost of the system which consists of the license costs and the transportation cost over the network. Each

server can handle only jobs up to the limit of its CPU, this is controlled by equation (2). Equation (3) states that each customer job must be handled by exactly one server. The last constraint (4) states that if a customer application is assigned to a server, then the corresponding application must be provided on the server.

Looking at the constraints we see that this model is an extended version of the capacitated warehouse location problem Mirchandani and Francis [1990]. This is related to the (simpler) uncapacitated warehouse location problem described in Van Hentenryck and Carillon [1988], one of the first models developed with finite domain constraint programming. Not surprisingly, the problem in Chiopris and Fabris [1994] was also expressed using the finite domain solver of CHIP Dincbas et al. [1988]. It is doubtful whether for this problem the approach is competitive with state of the art local search methods Michel and Van Hentenryck [2004].

Note that the network view of this model is quite simplistic. We only have to pay a transmission cost which depends on the choice of source and sink for each demand. We do not consider if there is enough network capacity to transport the traffic over the network, or which routes should be used. For the application placement model, network capacity is infinite. We will see in the following section how a more detailed view of the network could be used to add more constraints to the problem.

3.2 Path Placement

In many data networks we can decide which path is used for a demand, and a central control algorithm is responsible for assigning paths to demands under the capacity constraints of the network. There are three main alternative models for solving this problem, they are called

link-based For each demand we have one decision variable per link which states if the link is used for this demand or not.

path-based For every demand we have one decision variable per possible path between source and destination. We can choose one path per demand.

node-based For every demand we have a decision variable for every node in the network. If its value is 0, then the node is not used by the demand, if it is non-zero, then it gives the successor node. The variables for each demand form a cycle in the graph.

We also present two variants of the path placement problem in this section.

- In the *demand acceptance problem*, we are given a set of demands and have to decide which demands we can accept without exceeding the capacity limits of the network. We are interested in finding a solution which maximizes the value of the accepted demands.
- The *traffic placement problem* uses a fixed set of demands, which all have to be placed. We search for a solution which minimizes the maximal utilization over all network links.

The base-line comparison for these problems is the *CSPP (Constrained Shortest Path First)* algorithm Davie and Rekhter [2000], which applies a simple greedy heuristic to place the demands in order of decreasing demand size. For each demand we attempt to place it on the network using a shortest path algorithm in a residual graph, where only edges with more than the required bandwidth are considered. If the demand can be placed, it is accepted, and the free capacity in the network is reduced by its demand size.

We now discuss the different models in more detail, using the following conventions. The network consists of a set of nodes \mathbf{N} and a set of directed edges \mathbf{E} . The set $\mathbf{OUT}(n)$ consists of all edges leaving node n , the set $\mathbf{IN}(n)$ of all edges leading to node n . The capacity of an edge is given by $\mathbf{cap}(e)$. The source of an edge is denoted as $\mathbf{source}(e)$, the sink as $\mathbf{sink}(e)$. For every demand in the set \mathbf{D} , we have a source ($\mathbf{orig}(d)$) and a destination ($\mathbf{dest}(d)$), a bandwidth requirement $\mathbf{bw}(d)$ and a value $\mathbf{val}(d)$ which indicates the benefit of accepting demand d .

Link-based Model We start with the *demand acceptance problem* in a link-based model, where we use one decision variable for every demand and link of the network. The $\{0, 1\}$ variable X_{de} denotes whether demand d is routed over edge e of the network. For every demand d we also have one $\{0, 1\}$ decision variable Z_d which indicates if the demand is accepted or not. The following model gives a MILP formulation.

$$\max_{\{Z_d, X_{de}\}} \sum_{d \in \mathbf{D}} \mathbf{val}(d) Z_d \quad (5)$$

st.

$$\forall d \in \mathbf{D}, \forall n \in \mathbf{N} : \sum_{e \in \mathbf{OUT}(n)} X_{de} - \sum_{e \in \mathbf{IN}(n)} X_{de} = \begin{cases} -Z_d & n = \mathbf{dest}(d) \\ Z_d & n = \mathbf{orig}(d) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

$$\begin{aligned} \forall e \in \mathbf{E} : \quad & \sum_{d \in \mathbf{D}} \mathbf{bw}(d) X_{de} \leq \mathbf{cap}(e) & (7) \\ & Z_d \in \{0, 1\} \\ & X_{de} \in \{0, 1\} \end{aligned}$$

The *path constraint* (6) states that there is a single path for each accepted demand, linking the Z_d and X_{de} variables. Equation (7) states the *capacity constraint*, that for each link the amount of traffic routed over it must be smaller than the link capacity. The objective (5) is to maximize the value of the accepted demands.

We often also have *quality of service* constraints attached to the demands. For each demand d we have a quality requirement $\mathbf{req}(d)$ which should not be exceeded in our solution. The quality of the assigned path is calculated as the sum of constants $\mathbf{del}(e)$ for all edges which are used by the path. A typical

quality indicator is delay (also called *latency*), which is calculated as the sum of the propagation delays of the links on the selected path. This delay must be smaller than the latency requirement expressed for the demand. The constraints take the form given in equation (8). If the required latency is too small, then we may not be able to find any path which satisfies the demand.

$$\forall d \in \mathbf{D} : \sum_{e \in \mathbf{E}} \text{del}(e) X_{de} \leq \text{req}(d) \quad (8)$$

The *traffic placement problem* is expressed in a similar form. Here all demands must be accepted, we therefore no longer need Z_d variables. The only variables in the model are $\{0, 1\}$ variables X_{de} which indicate whether demand d is routed over edge e . We still have the *path constraint* (equation 10) and an objective function (9) which is to minimize the maximal relative utilization of any edge in the network. For this we divide the amount of traffic routed over the edge by the capacity of the edge. A solution with a cost of less than 1 means that the traffic routed over all edges stays within the link capacity, if the cost is greater than 1, then some links exceed their capacity.

$$\min_{\{X_{de}\}} \max_{e \in \mathbf{E}} \frac{1}{\text{cap}(e)} \sum_{d \in \mathbf{D}} \text{bw}(d) X_{de} \quad (9)$$

st.

$$\forall d \in \mathbf{D}, \forall n \in \mathbf{N} : \sum_{e \in \mathbf{OUT}(n)} X_{de} - \sum_{e \in \mathbf{IN}(n)} X_{de} = \begin{cases} -1 & n = \text{dest}(d) \\ 1 & n = \text{orig}(d) \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

$$X_{de} \in \{0, 1\}$$

In this model the capacity constraints (7) are folded into the objective function, they will appear as linear constraints in a MILP representation. Unfortunately, there seems to be no consensus about the objective function. There are at least two other versions proposed in different papers. One is to minimise the overall network utilization

$$\min_{\{X_{de}\}} \sum_{e \in \mathbf{E}} \sum_{d \in \mathbf{D}} \text{bw}(d) X_{de} \quad (11)$$

while imposing the capacity constraints (7). This objective function is closely related to the formulation of the *Integer Multi-Commodity Flow Problem* Ahuja et al. [1993]. The other alternative is proposed in Ouaja and Richards [2003] as a compromise which is supposed to return better overall solutions. It minimizes the average link utilization.

$$\min_{\{X_{de}\}} \sum_{e \in \mathbf{E}} \sum_{d \in \mathbf{D}} \frac{\text{bw}(d)}{\text{cap}(e)} X_{de} \quad (12)$$

Overall, demand acceptance and these versions of traffic placement are closely related, but their objectives are different enough to make a direct comparison between results difficult. In later section, we will only describe the demand acceptance problem, the traffic placement variant can be derived in a similar way.

We now discuss a number of proposed algorithms that use the link based model.

Lagrangian Relaxation/CP Hybrid - Path decomposition A hybrid method for traffic placement is proposed in Ouaja and Richards [2003, 2004]. It combines *Lagrangian Relaxation (LR)* (see chapter 15, “Operations Research Methods in Constraint Programming”) with constraint programming. Both models are set up in parallel and exchange information during a depth-first search. In the Lagrangian subproblem we dualize the capacity constraints (7) into the cost function and keep the path constraints (6).

The constraint model uses finite domain $\{0, 1\}X_{de}$ variables and finite domain versions of the path constraints, strengthened by some redundant inequality constraints to remove cycles. In a preprocessing step, some capacity constraints based on s-t cuts are added. The LR model starts with an initial heuristic, possibly infeasible assignment obtained with a CSPF variant.

At each node of the search tree, a limited number of subgradient steps Ahuja et al. [1993] are performed in the LR model. This involves solving $|\mathbf{D}|$ shortest path problems with an LP solver. The results are used to generate new constraints for the FD model, which, through propagation, may lead to further assignments, which are then returned to the LR model. When the LR model is stopped, the current Lagrangian solution is used to decide on the next branching choice, assigning some undecided X_{de} variable. Nodes in the search can be pruned either because the LR model becomes infeasible or because constraint propagation detects a failure.

The results obtained in Ouaja and Richards [2003, 2004] can not be directly compared with other methods, since the model uses a different cost function (12). But Ouaja and Richards [2005], Cronholm and Ajili [2005] describe experiments with a modified version of this decomposition where the more commonly accepted cost function (11) is used.

Lagrangian Relaxation/CP Hybrid - Knapsack decomposition As an alternative Lagrangian Relaxation Ouaja and Richards [2005] proposes to relax the path constraints (6) while keeping the capacity constraints (7). This means that at each subgradient step, we have to solve $|\mathbf{E}|$ independent knapsack ILP problems.

Both Ouaja and Richards [2003] and Ouaja and Richards [2005] use cost-based filtering Focacci et al. [1999] from the reduced cost of the X variables of the LP relaxation solved at each subgradient step to fix some values to either 0 or 1. This idea is extended in Cronholm et al. [2004, 2005], which adds three more filtering rules. Comparative experiments show that this produces stronger pruning, which improves the quality of the solutions, reduces the number of nodes in the search tree significantly and can lead to overall savings in computation time.

Probe Backtracking Probe Backtracking Search (PBT) El Sakkout and Wallace [2000] is used in Liatsos et al. [2003] to solve the traffic placement problem with a link-based model. The idea behind PBT is to split the constraints into two groups, considered "easy" and "hard". The hard constraints and the optimization are handled by a branch&bound backtracking scheme. In each node, one of the hard constraints is replaced by a disjunction of easy constraints, which form the basis of the branching decisions. The system starts with a tentative assignment based on a heuristic. In each node the current set of easy constraints is used to solve a subproblem called *prober* which either detects infeasibility or returns a new tentative, partial assignment. Violations of the hard constraints are resolved by branching on the introduction of additional simple constraints.

For the traffic placement problem the prober finds a cycle-free path for one demand which respects the delay constraint (8) and any imposed forbidden and required links. The prober is implemented as an incremental ILP which adds no-good cuts until a valid solution is returned. The search component checks for violations of the capacity constraints; if there are none, a solution has been found. Otherwise, it heuristically selects a link with a capacity violation and a demand routed over it, and calls the prober again, branching on either forbidding or enforcing the selected link for the selected demand.

The probe backtracking scheme can be used with a variety of probing methods, Kamarainen [2003] considers the combination with local search, and Kamarainen and El Sakkout [2004] applies this combination to demand acceptance. The local probe consists of a simulated annealing local search routine which tries to find paths for a set of demands. One of the challenges for the implementation lies in how to incorporate forced links into the neighborhood operator.

Path-based Model The second type of formulation for the path placement problem is the path-based model. In this model we consider for each demand the paths on which it can be routed. Assume that there are $\text{path}(d)$ possible paths for demand d . We introduce a $\{0, 1\}$ decision variable Y_{id} for each possible path for each demand d . The constants h_{id}^e indicate whether path i for demand d is routed over edge e . The $\{0, 1\}$ variables Z_d again state whether demand d is accepted or not.

$$\max_{\{Z_d, Y_{id}\}} \sum_{d \in \mathbf{D}} \text{val}(d) Z_d \quad (13)$$

st.

$$\forall d \in \mathbf{D} : \sum_{1 \leq i \leq \text{path}(d)} Y_{id} = Z_d \quad (14)$$

$$\forall e \in \mathbf{E} : \sum_{d \in \mathbf{D}} \text{bw}(d) \sum_{1 \leq i \leq \text{path}(d)} h_{id}^e Y_{id} \leq \text{cap}(e) \quad (15)$$

$$Z_d \in \{0, 1\}$$

$$Y_{id} \in \{0, 1\}$$

The objective function (13) is the same as for the link based model (5). Equation (14) links the Z_d and the Y_{id} variables and states that at most one path may be selected for each demand. Equation (15) is a modified capacity constraint, stating that the bandwidth required for all selected paths routed over an edge must be smaller than the edge capacity.

Column Generation The direct implementation of a path based model will be difficult due to the very large number of possible paths even in a small graph. This model therefore is a natural candidate for techniques like column generation, where only a limited subset of all possible paths are considered and the objective function can drive the search for finding new paths which improve overall solution quality.

We will discuss this approach used for example in Chabrier [2003] in more detail in section 3.5.

Blocking Islands A different, more CSP oriented view of a path-based model is given in Frei and Faltings [1999]. It considers the traffic placement problem without objective function, i.e. all demands must be placed, but we only require a feasible solution. This can be modelled as a CSP where each demand is a variable which ranges over all possible paths. The main idea of this paper is that it is not necessary to describe the domains of the variables explicitly, it is sufficient if we can check whether there still is a possible path for each demand at each step. This corresponds to a forward-checking based constraint propagation, which can be used inside a search routine that assigns paths to demands. Heuristics like dynamic variable and value selection can also be applied, making this a nice example of an implied constraint model.

Key to the implementation of this implied model is the concept of a *blocking island*. For a given demand size d , we can partition the nodes in the network into equivalence classes, called d blocking islands. All nodes inside one blocking island can reach each other with paths of at least size d . We know that a demand of size d can be satisfied if both source and sink are in the same d island, and can not be satisfied if they are in different d islands. The blocking islands for different capacity values form a tree, where the root is one island connecting all nodes with capacity 0.

When a path is assigned to a demand, we can update the blocking island tree effectively, and can check if there still is a path for every demand in the new tree. If not, the node in the search tree can be pruned, and we backtrack to a previous choice. The experiments in Frei and Faltings [1999] showed that it can be worthwhile to replace chronological backtracking with a form of conflict-driven backjumping.

The approach can be easily extended to handle the objective function for traffic placement, but it is unclear how to adapt this technique to the demand acceptance problem.

Local Search/CP Hybrid Another hybrid solver for the demand acceptance problem is described in Lever [2004, 2005]. It combines local search with finite domain

constraint programming in a multi-step procedure, and also combines elements of a path based and a link based model. It starts with a heuristic, CSPF based initial selection of paths to be chosen. In a second step, it tries to add additional demands one by one, at each step calling a local search based repair method to remove any capacity bottlenecks. This repair works by replacing an existing path with a new one. As a third step it uses a hybrid branch&bound routine combining finite domains with local search. The finite domain model initially only consists of the acceptance variables Z_d , the X_{de} variables are only lazily generated on demand. The system finds necessary links for a demand, and imposes capacity constraints on them. It also considers certain cut sets and creates X_{de} variables for each of those links. It then adds a constraint stating that their sum must be equal to the demand variable Z_d as well as the link capacity constraints over all demands on that link.

The local search routine identifies capacity violations and builds a neighborhood by selecting demands currently routed through a violation, and choosing a new shortest path for each demand, using the capacity violation as link metric. The neighborhood is evaluated based on the improvement of the capacity violations, and a move selected using a randomized choice which allows some decrease in quality for some moves. A restart method was incorporated to go back to a previous state if no improvement had been achieved within a given number of steps.

A comparison of this method with a variant of Liatsos et al. [2003] for demand acceptance showed that both were complementary, each method was more successful on some problem instances, while both clearly outperformed CSPF.

Node-based Model In the node-based model the decision variables define successor relations between network nodes, defining paths through a network via the nodes that are traversed. This model is due to Ros et al. [2001], which describe this model in terms of a *Bandwidth on Demand* application (see section 3.4). For each demand d and each node k in the network, we introduce an integer decision variable S_{kd} with the following domain:

$$S_{kd} :: \begin{cases} \{\mathbf{sink}(e) | e \in \mathbf{OUT}(k)\} & k = \mathbf{orig}(d) \\ \mathbf{orig}(d) & k = \mathbf{dest}(d) \\ \{0\} \cup \{\mathbf{sink}(e) | e \in \mathbf{OUT}(k)\} & \text{otherwise} \end{cases} \quad (16)$$

For each demand the domain for a node contains all possible successors and the value 0, which indicates that the node is not used to route the demand. We add a back-link from the destination of the demand to the source, which does not correspond to an edge in the graph. We now require that for every demand d the set

$$\{ \langle k, S_{kd} \rangle \mid S_{kd} \neq 0 \} \quad (17)$$

forms a cycle in the graph (augmented with the back link). In Ros et al. [2001], this condition is expressed with the *cycle* constraint Beldiceanu and Contejean [1994], Bourreau [1999] of CHIP. This constraint does not allow conditional

nodes, the model therefore needs dummy nodes and edges to connect all unused nodes in a second cycle.

The capacity constraint for each node can be expressed with a cumulative constraint Aggoun and Beldiceanu [1993], which uses two arguments, a set of tasks given by *start*, *duration* and *resource use* and a resource profile, given as a set of tuples *time point* and *resource limit*.

$$\text{cumulative}(\{ \langle S_{id}, 1, \text{bw}(d) \rangle \mid d \in \mathbf{D} \}, \{ \langle l, m \rangle \mid 0 \leq l \leq n, m = \begin{cases} \infty & l = 0 \\ \text{cap}(e) & \exists e \in \mathbf{Est}. \text{source}(e) = i, \text{sink}(e) = j \\ 0 & \text{otherwise} \end{cases} \}) \quad (18)$$

In this model we need $|\mathbf{D}|$ cycle constraints and $|\mathbf{N}|$ cumulative constraints to express the conditions of the routing problem, we then have to define a search routine and a `min_max` Prestwich [1999] optimization routine to find the optimal solution. Additional conditions like quality of service constraints can be handled by using extra arguments of the cycle constraints.

The cycle constraint probably is not the best choice for this application, the need for dummy nodes and edges in the model destroys much of its propagation potential.

3.3 Multiple Paths

In the models of section 3.2 we have looked for a single path for each demand. This path is used to transmit the traffic between the source and the destination of the demand. What happens if one of the elements on the path fails? There are three possible scenarios:

- The transmission for the demand is interrupted until the element failure is repaired. This may lead to an outage of several hours and is normally not acceptable.
- We dynamically search for a new path in the modified network and set up the path for transmission. The outage will be much shorter, and is limited by the time required to find a new path.
- We have pre-computed an alternative path, which is *link disjoint* to the original path, so that the element failure does not affect this additional, secondary path. We can immediately switch to the alternative, minimizing the outage.

We will now look at the problem of finding multiple (primary and secondary) paths for demands, where the primary path is used in normal operation, and the secondary is only active when some link on the primary path has failed. We present a link-based, demand acceptance model for this problem, which is an extension of the model in section 3.2. In addition to the $\{0, 1\}$ variables Z_d for demand acceptance, and X_{de} for the primary path, we need additional decision

variables W_{de} , which indicate whether edge e is used for the secondary path of demand d .

$$\max_{\{Z_d, X_{de}, W_{de}\}} \sum_{d \in \mathbf{D}} \text{val}(d) Z_d \quad (19)$$

st.

$$\forall d \in \mathbf{D}, \forall n \in \mathbf{N} : \sum_{e \in \mathbf{OUT}(n)} X_{de} - \sum_{e \in \mathbf{IN}(n)} X_{de} = \begin{cases} -Z_d & n = \text{dest}(d) \\ Z_d & n = \text{orig}(d) \\ 0 & \text{otherwise} \end{cases} \quad (20)$$

$$\forall e \in \mathbf{E} : \sum_{d \in \mathbf{D}} \text{bw}(d) * X_{de} \leq \text{cap}(e) \quad (21)$$

$$\forall d \in \mathbf{D}, \forall n \in \mathbf{N} : \sum_{e \in \mathbf{OUT}(n)} W_{de} - \sum_{e \in \mathbf{IN}(n)} W_{de} = \begin{cases} -Z_d & n = \text{dest}(d) \\ Z_d & n = \text{orig}(d) \\ 0 & \text{otherwise} \end{cases} \quad (22)$$

$$\forall e \in \mathbf{E}, \forall e' \in \mathbf{E} \setminus e : \sum_{d \in \mathbf{D}} \text{bw}(d) * (X_{de} - X_{de'} * X_{de} + X_{de'} * W_{de}) \leq \text{cap}(e) \quad (23)$$

$$\forall d \in \mathbf{D}, \forall e \in \mathbf{E} : X_{de} + W_{de} \leq 1 \quad (24)$$

$$Z_d \in \{0, 1\}$$

$$X_{de} \in \{0, 1\}$$

$$W_{de} \in \{0, 1\}$$

The model is quite similar to the model in section 3.2. The objective function (19) is the same, and we also find the path constraint (20) and the capacity constraints (21) for the primary path. To this we add a path constraint for the secondary path (equation 22) and the link-disjoint constraint (24) which states that primary and secondary paths for a demand can not share a link. The only complicated, additional constraint is the capacity constraint for the secondary paths (equation 23). This expresses the capacity for link e in the case of failure of link e' . The traffic we have to consider is the sum of all primary paths through link e , except for those that were also routed through link e' . In addition we need to consider all secondary paths routed through e , where the primary was routed through the failed link e' . We therefore need a capacity constraint for each link under each failure scenario. This dramatically increases the number of constraints that are required.

The link disjoint constraint is only one way of keeping the primary and secondary paths apart. We can also consider two stronger alternatives, the paths may be *node disjoint* or *SRLG disjoint*.

For *node disjoint* primary and secondary paths we enforce that, except for the source and destination nodes, the paths do not cross the same nodes. This protects the connection even in case of a node failure. We can express this constraint

by the equations

$$\forall d \in \mathbf{D}, n \in \mathbf{N} \setminus \{\mathbf{orig}(d), \mathbf{dest}(d)\} : \sum_{e \in \mathbf{IN}(n)} X_{de} + W_{de} \leq 1 \quad (25)$$

$$\forall d \in \mathbf{D}, n \in \mathbf{N} \setminus \{\mathbf{orig}(d), \mathbf{dest}(d)\} : \sum_{e \in \mathbf{OUT}(n)} X_{de} + W_{de} \leq 1 \quad (26)$$

Using either (25) or (26) together with the path constraint is enough to ensure the condition.

The *SRLG disjoint* constraint is a generalization to arbitrary sets of links. A *shared risk link group (SRLG)* is a set of links that we consider might fail together. This can happen for example if several (physical) links are run through the same cable duct, or if multiple logical (layer-3) connections are mapped to the same physical (layer-1) connections, for example multiple carriers using the same sea-cable for a intercontinental connection. To express this constraint, we assume we are given a set \mathbf{S} of SRLG sets. For each SRLG \mathbf{SRLG} and each demand d , we express the constraint that primary and secondary path can not share any links in the set.

$$\forall d \in \mathbf{D}, \forall \mathbf{SRLG} \in \mathbf{S} : \sum_{e \in \mathbf{SRLG}} X_{de} + W_{de} \leq 1 \quad (27)$$

The model above is the basis for the solver described in Xia and Simonis [2005]. A linearization of the non-linear constraints (23) adds a many new variables and linear constraints. To handle this very large number of capacity constraints required for the failure cases the authors suggested the following decomposition. Initially, the problem is set up without constraints (23) and solved with a MILP solver. For the optimal solution the secondary path capacity constraints are checked and, if violated, added (linearized) to the constraint set. The process iterates until no secondary path capacity constraints are violated, in which case an optimal solution to the whole problem is found. This can be seen as a form of *Benders decomposition* (see chapter 15, “Operations Research Methods in Constraint Programming”), where the secondary path capacity constraints form the sub problems, and all other constraints form the master problem. The *Benders cuts* that are generated take the form of the linearized capacity constraints. This approach allowed a significant speed-up over a standard MILP formulation of the problem.

3.4 Bandwidth on Demand

So far, the traffic demands we encountered were all for a single snapshot in time, i.e. all demands were simultaneous. We now consider the *Bandwidth on Demand* scenario where each demand has a fixed start $\mathbf{start}(d)$ and end $\mathbf{end}(d)$ time, and demands only interact if they overlap in time. The problem is to accept demands so that at no time point the capacity of the network is exceeded. Instead of looking at every time point between the earliest start and the latest

end of any demand, we can restrict ourselves to time points when new demands are starting. The set of time points \mathbf{T} to consider is then given by

$$\mathbf{T} = \{\text{start}(d) | d \in \mathbf{D}\} \quad (28)$$

We can now formulate the demand acceptance problem for Bandwidth on Demand with the following MILP, which uses the Z_d and $X_{de} \in \{0, 1\}$ decision variables already familiar from section 3.2.

$$\max_{\{Z_d, X_{de}\}} \sum_{d \in \mathbf{D}} \text{val}(d) Z_d \quad (29)$$

st.

$$\forall d \in \mathbf{D}, \forall n \in \mathbf{N} : \quad \sum_{e \in \text{OUT}(n)} X_{de} - \sum_{e \in \text{IN}(n)} X_{de} = \begin{cases} -Z_d & n = \text{dest}(d) \\ Z_d & n = \text{orig}(d) \\ 0 & \text{otherwise} \end{cases} \quad (30)$$

$$\forall t \in \mathbf{T}, \forall e \in \mathbf{E} : \quad \sum_{\substack{d \in \mathbf{D} \\ \text{start}(d) \leq t \\ t < \text{end}(d)}} \text{bw}(d) X_{de} \leq \text{cap}(e) \quad (31)$$

$$Z_d \in \{0, 1\}$$

$$X_{de} \in \{0, 1\}$$

We find the objective function (29), the path constraint for each demand (30) and a modified capacity constraint (equation 31). This capacity constraint is now quite similar to the global cumulative constraint Aggoun and Beldiceanu [1993]. For every time point and every link, the sum of the accepted traffic routed over the link may not exceed the link capacity.

There are many ways to extend the basic Bandwidth on Demand model. Simple modifications are changes of the link capacity over time or general topology changes at fixed time points. We may also consider that demands may be moved from one path to another during their life-time, in order to accommodate changes in the demand structure.

Another natural extension is some form of on-line algorithm, where new demands for future time periods are added from time to time. Previous commitments must be respected, i.e. demands that were accepted at an earlier time point can not be rejected later on. This is easily handled by forcing some of the Z_d variables to be 1.

A much more challenging extension is to relax the fixed start and end times. If demands can be moved in time, we obtain a very difficult combination of the path placement and a cumulative scheduling problem.

France Telecom The problem first described in Lauvergne et al. [2002], and then reconsidered in Loudni et al. [2003] is an on-line version of the Bandwidth on

demand problems for ATM networks. When a new demand is entered, we have to check if it can be accepted, possibly by rerouting previously accepted demands. There is a time limit of one minute to decide on acceptance. But once a demand has been accepted, it can not be rejected at a later point in time.

The model of Lauvergne et al. [2002] uses a constraint based conflict resolution mechanism which is called when a demand can not be placed on top of the existing set of connections. It selects short paths for the new demands and finds all existing connections which would be in conflict when accepting the new demand on one of the routes. It then creates a constraint model which tries to repair the solution by moving one of the existing connections to a new path. The resulting capacity checks are quite complex, since they have to be performed for every time point between the start and the end of the task considered. A specialized calendar data structure is used to minimize the overhead.

The constraint model for the path constraint takes an unusual form in this model. It is based on an assignment from a n-th hop variable to the links of the network. Therefore, the length of each path must be bounded a priori, and a dummy value for “not used” must be introduced.

The procedure is compared to the on-line CSPF algorithm, which does not consider rerouting of existing connections, and achieves some improvement in the percentage of accepted demands.

The approach in Loudni et al. [2003] uses the same basic problem formulation, but uses *Valued CSP (VCSP)* (see chapter 9, “Soft Constraints”) as the conflict resolution mechanism. The VCSP is solved by a combination of finite domain, local search and limited discrepancy search Harvey and Ginsberg [1995].

Schlumberger dexa.net The Bandwidth on demand system for Schlumberger’s dexa.net Symes [2004] decomposes the problem on a temporal basis. For each time point, a specialized solver for the demand acceptance problem is run, its results form new constraints for the next time point.

This technique is expanded in Chu and Xia [2005], which proposes three alternative models. In the first one, all orders starting at the same time are processed together, with different events being treated sequentially. In the second model, the overall planning horizon is split into intervals, and all tasks starting in that interval are planned together. In the last model, a Benders decomposition is used to link subproblems for each interval together in the master problem. Tasks which extend over several intervals cause inter-dependency between the sub problems, the generated Benders cuts guide the procedure to the optimal solution. The first two methods are incomplete, the third one is complete. Experiments and the comparison with a MILP formulation show that the temporal decomposition is much faster than, but can not reach the quality of, the Benders decomposition, which in turn is similar to the MILP solution in quality, while requiring significantly less computational effort.

Global Constraint Model As described in section 3.2, the approach of Ros et al. [2001] uses a node-based model. To handle the time dimension of the Bandwidth

on Demand problem, the authors replace the cumulative constraints (18) with a four dimensional diffn constraint Beldiceanu and Contejean [1994].

3.5 Network Design and Capacity Planning

The problems in sections 3.2 to 3.4 were all considering a fixed network structure. The topology of the network and the link capacities were given, the variables were introduced by demands and the overall utilization of the network. When planning to build a network, a different question arises: How should I connect the nodes in my network, and which capacity should I use for each link? A similar problem is considered in capacity planning. Given the current network and set of projected demands, how should I extend the network to cope with future demands. We now consider this problem, following the problem specification in Le Pape et al. [2002]. For every potential edge in the network, we have a set of $\text{alt}(e)$ possible design alternatives with bandwidth $\text{cap}(i, e)$ and cost $\text{cost}(i, e)$. One of these alternatives might be not to use that link. This alternative may have non-zero cost in the case of capacity planning, when the link already exists and we have to pay a de-commissioning cost in order to remove it. In our model we have two types of variables, the $X_{de} \{0, 1\}$ variables which indicate whether a link is used by a demand, and W_{ie} decision variables which indicate if design alternative i for edge e is chosen.

$$\min_{\{X_{de}, W_{ie}\}} \sum_{e \in \mathbf{E}} \sum_{1 \leq i \leq \text{alt}(e)} \text{cost}(i, e) W_{ie} \quad (32)$$

st.

$$\forall d \in \mathbf{D}, \forall n \in \mathbf{N} : \quad \sum_{e \in \text{OUT}(n)} X_{de} - \sum_{e \in \text{IN}(n)} X_{de} = \begin{cases} -1 & n = \text{dest}(d) \\ 1 & n = \text{orig}(d) \\ 0 & \text{otherwise} \end{cases} \quad (33)$$

$$\forall e \in \mathbf{E} : \quad \sum_{d \in \mathbf{D}} \text{bw}(d) X_{de} \leq \sum_{1 \leq i \leq \text{alt}(e)} \text{cap}(i, e) W_{ie} \quad (34)$$

$$\forall e \in \mathbf{E} : \quad \sum_{1 \leq i \leq \text{alt}(e)} W_{ie} = 1 \quad (35)$$

$$W_{ie} \in \{0, 1\}$$

$$X_{de} \in \{0, 1\}$$

The objective function (32) is to minimize the total cost of the design. Equation (33) states the usual path constraint for each demand. The design choice constraints (35) state that for each edge we have to choose one alternative (which might be not to use the link). The capacity constraint (34) compares the traffic volume routed over the link with the capacity provided by the chosen design alternative.

ROCOCO benchmarks Reference Le Pape et al. [2002] provides an evaluation of five alternative solution methods for a set of benchmark problems from France Telecom, and also introduces combinations of additional constraints for this problem to create a broader set of test cases. It looks at a CP, a MILP and a column generation alternative in an attempt to improve first results, and further develops the constraint model into a hybrid with local search and parallel execution on multiple processors. An initial heuristic solution is created with CP, a local search module then tries to improve the solution found, and finally a tree search is started with the best current solution as an upper bound.

The problem sizes range from very small (4-10 nodes) to medium size (15-25 nodes), and a timeout of 10 minutes is imposed. For a design problem that is a very small execution time, this might be explained by the large number of problem variations being tested. Over the implemented variants, the CP+local search routine is the most effective, but unfortunately details of the different models are quite sparse.

The model used by Chabrier [2003] is path based, using column generation in a branch and price and cut framework. The paper describes various cutting planes, which strengthen the basic branch and price framework, which alone is not competitive. But even with the cuts added, the system does not find solutions within the timeout using the default search method. A custom search routine based on limited discrepancy search Harvey and Ginsberg [1995] was used to find solutions, results indicated that it was performing better than Le Pape et al. [2002].

Design for Multicast In Cronholm and Ajili [2004, 2005] a variant of the design problem is studied. Instead of designing a network for point to point demands, the authors consider multi-cast traffic demands. In multi-cast, a traffic source injects traffic to the network which must be delivered to a number of subscribers. The traffic is routed over a tree rooted in the traffic source, with consumers located on the leaf nodes. Multi-cast traffic on IP networks is becoming more and more significant, as it is a much cheaper form of content distribution for applications like video-on-demand, broadcasting, or large scale software updates. The two papers differ in the method used, Cronholm and Ajili [2004] describes a hybrid of Lagrangian relaxation with constraint programming, while Cronholm and Ajili [2005] considers a hybrid of a branch-and-price column generation with a finite domain constraint component.

SONET Network Design Although also a network design problem, the problem studied in Smith [2005] bears little relation to the model shown above. Smith [2005] shows results for using Constraint Programming on a design problem of a SONET/SDH optical network. Instead of point-to-point links, SONET networks use fiber rings to which a number of users can be connected via hardware devices. The ring capacity can be filled with traffic between users on the same ring. The objective is to choose the minimal number of rings and of hardware interfaces to satisfy all communication demands.

3.6 IGP Metric Optimization

The problems considered in sections 3.2 to 3.4 were all about explicit traffic placement, i.e. a central control algorithm computed single paths for demands which together satisfied the global capacity constraints. In marked contrast stand networks that use routing and packet switching. Here each packet is forwarded locally not by following a fixed connection between source and destination, but by local routing decisions which control at each node where the packet is sent next. A distributed routing algorithm is used so that each router makes its forwarding decisions only based on local knowledge, and control messages are sent between the nodes to inform them about the overall topology and changes to the connectivity. The routing protocols are designed to converge to a consistent routing after each change within a limited time period, the *re-convergence time*. Routing inside the network is controlled by the *Interior Gateway Protocol (IGP)*, routing to destinations outside the current network is controlled by the *Border Gateway Protocol (BGP)* Peterson and Davie [2000]. There are many variants of IGP routing protocols (like RIP, OSPF Moy [1998], IS-IS, EIGRP) Malhotra [2002], which mainly differ in the way the distributed network nodes exchange information and what global view of the network is maintained in each node.

From the outside, IGP routing in a steady state can be seen as an application of shortest path finding. Each link in the network is assigned an edge weight, the *routing metric*, and traffic through the network follows a shortest path between source and destination. By changing the routing metric we can (indirectly) control which paths are chosen and how much traffic can be placed on the network.

An important question is what happens if multiple shortest paths exist between two nodes. Depending on the protocol and its configuration, the system may

- select one of the paths at random
- split traffic between multiple alternatives in a balanced way
- split traffic between multiple alternatives in arbitrary fashion

There typically is also a hardware-based limit on how many (typically 8 or 16) shortest path alternatives can be handled in each node.

All this makes it very hard to predict where traffic will be placed when multiple shortest paths exist. Some of the systems for IGP metric optimization therefore enforce a constraint that only a single shortest path can exist between any two nodes in the network.

We now present an example of a path-based model for IGP metric optimization, which enforces the single shortest path rule. The model uses positive integer (not $\{0, 1\}$) variables W_e for the edge weights. We have $\{0, 1\}$ variables Y_{id} for each demand d and each of the $\text{path}(d)$ possible paths for the demand, which indicate whether this path is used. We also introduce continuous variables P_{id} which describe the total weight of the path i for demand d as the sum of the weights of the edges traversed. The $\{0, 1\}$ constants h_{id}^e indicate whether path i

for demand d traverses edge e .

$$\min_{\{Y_{id}, W_e\}} \max_{e \in \mathbf{E}} \frac{1}{\text{cap}(e)} \sum_{d \in \mathbf{D}} \text{bw}(d) \sum_{1 \leq i \leq \text{path}(d)} h_{id}^e Y_{id} \quad (36)$$

st.

$$\forall d \in \mathbf{D} : \sum_{1 \leq i \leq \text{path}(d)} Y_{id} = 1 \quad (37)$$

$$\forall d \in \mathbf{D}, 1 \leq i \leq \text{path}(d) : P_{id} = \sum_{e \in \mathbf{E}} h_{id}^e W_e \quad (38)$$

$$\forall d \in \mathbf{D}, 1 \leq i, j \leq \text{path}(d) : P_{id} = P_{jd} \implies Y_{id} = Y_{jd} = 0 \quad (39)$$

$$\forall d \in \mathbf{D}, 1 \leq i, j \leq \text{path}(d) : P_{id} < P_{jd} \implies Y_{jd} = 0 \quad (40)$$

$$Y_{id} \in \{0, 1\}$$

$$\text{integer } W_e \geq 1$$

$$P_{id} \geq 0$$

The objective function (36) is to minimize the maximal relative utilization of any link in the network. Constraint (37) enforces that there is a single, selected shortest path. The weight of each path is controlled by equation (38) as the sum of the weights of the traversed edges. Constraint (39) states that paths for one demand with the same weight must have identical Y_{id} values, which must be zero, as only one shortest path is allowed. Equation (40) states that a path will not be selected if a shorter one exists. Note that this is not intended as an effective model to generate weights, but serves as a declarative problem specification. We now discuss a number of solution approaches to this problem:

Branch and price A complete method for the unique path weight setting problem is given in Ajili et al. [2005a]. It is based on a path-based model using column generation in a branch and price framework using a hybrid with finite domain constraints. Paths for demands are created lazily using the pricing information from the master problem as a guide.

Preliminary experimental results indicate that the hybrid outperforms more straightforward MILP models, but has difficulty scaling to larger problem instances. One problem is the lack of a good initial solution which can be used as a starting point for the tree search.

Tabu search A combination of a tabu based local search and a MILP is presented in Ajili et al. [2005b]. It is based on the observation that if a weight setting with fractional values achieves unique paths, then it can be easily converted into a weight setting using integer weights. So a tabu-based local search is first used to derive fractional weight settings, with the MILP converting the result into (small) integer weights. To come up with unique paths in the local search, an ingenious weight assignment is used which assigns weights 2^{-i} to the edges with

a different exponent i for each edge. Therefore each path in the network has a unique weight, and the search consists in finding a permutation of the edge assignments which minimizes the traffic load.

This algorithm scales well, although care must be taken not to cause rounding errors in the shortest path calculations due to the extreme range of weight values. Experimental results indicate that in the second part of the algorithm the weights can be re-assigned to integer values ranging from 1 to 200.

Set Constraint Solver A more generic problem is solved in Eremin et al. [2005]. The set based (see Chapter 17, “Beyond Finite Domains”) model presented in the paper allows to impose limits on the splitting of flows in each router. This can be used to express the condition mentioned above that the router hardware limits branching in a node to no more than m alternatives.

The model is a combination of set-valued variables describing the paths, continuous variables describing the flow volumes on each edge directed towards a node and integer variables for the edge weights and distance values. It is a good example for the use of different domain types within one model.

Experimental results show that feasible solutions which satisfy the branching limits are obtained easily, but that their quality is not close to the lower bound.

3.7 Flow Analysis and Resilience Analysis

So far we have assumed that as part of our problem definition we have a well-defined set of demands: We know who wants to use the network for connections between specific points and how much bandwidth they require. For IP based networks this assumption is not valid. In an operational network there is no (simple) way of collecting data about end-to-end traffic flows, we don't know who is talking to whom and how much bandwidth they use. The only information we can collect is the overall traffic on each edge on the network $\text{traf}(e)$ and the external traffic entering $\text{ext}^{in}(i)$ and leaving $\text{ext}^{out}(j)$ at each node of the network. We can try to reconstruct a demand matrix from these measurements, this is an active research area called *traffic flow analysis*.

A model for this problem is shown below. We use non-negative flow variables F_{ij} to denote the traffic flow from node i to node j in the network. The $[0, 1]$ constants r_{ij}^e define the routing in the network, they indicate what fraction of the flow between nodes i and j is routed over edge e .

$$\forall i, j \in \mathbf{N} : \quad \min_{\{F_{ij}\}} / \max_{\{F_{ij}\}} F_{ij} \quad (41)$$

st.

$$\forall e \in \mathbf{E} : \sum_{i,j \in \mathbf{N}} r_{ij}^e F_{ij} = \mathbf{traf}(e) \quad (42)$$

$$\forall i \in \mathbf{N} : \sum_{j \in \mathbf{N}} F_{ij} = \mathbf{ext}^{in}(i) \quad (43)$$

$$\forall j \in \mathbf{N} : \sum_{i \in \mathbf{N}} F_{ij} = \mathbf{ext}^{out}(j) \quad (44)$$

$$F_{ij} \geq 0$$

For every flow, we try to find a lower and an upper bound as the result of an optimization run with the objective (41). We know that the sum of all flows routed over an edge is equal to the observed traffic on the edge (42), and that the sum of all flows starting (43) or ending (44) in a node must be equal to the observed external traffic.

The fundamental problem with this approach is that it is very under-constrained. We have $|\mathbf{N}|^2$ flow variables F_{ij} , but only $|\mathbf{E}| + 2|\mathbf{N}|$ constraints. Results in Simonis [2003] show that the values for the flows can vary in a very wide interval, with no clear preference for any particular value. It is therefore unclear how to use the results for answering further questions about the network, for example how the traffic will change in case of an element failure.

The idea behind *resilience analysis* is to avoid the generation of the intermediate demand matrix, and to pose questions about the network behavior directly in the initial model. For example, we may be interested in understanding the traffic in the network under an element failure and resulting re-routing. The routing in the normal network operation is denoted with r_{ij}^e , the routing after the element failure is given by $r_{ij}^{\bar{e}}$. The model for resilience analysis below uses the flow variables F_{ij} only internally, without trying to deduce particular values.

$$\forall e \in \mathbf{E} : \min_{\{F_{ij}\}} / \max_{\{F_{ij}\}} \sum_{i,j \in \mathbf{N}} r_{ij}^{\bar{e}} F_{ij} \quad (45)$$

st.

$$\forall e \in \mathbf{E} : \sum_{i,j \in \mathbf{N}} r_{ij}^e F_{ij} = \mathbf{traf}(e) \quad (46)$$

$$\forall i \in \mathbf{N} : \sum_{j \in \mathbf{N}} F_{ij} = \mathbf{ext}^{in}(i) \quad (47)$$

$$\forall j \in \mathbf{N} : \sum_{i \in \mathbf{N}} F_{ij} = \mathbf{ext}^{out}(j) \quad (48)$$

$$F_{ij} \geq 0$$

The objective function (45) now tries to find a value for each edge in the network under the failure scenario, and finds bounds by running minimization and maximization optimization queries. The constraints (46, 47 and 48) are the same as for the traffic flow analysis.

Results in Simonis [2003] indicate that the bounds on the link traffic in failure scenarios are much tighter, and are close enough for most practical purposes.

In the discussion above, we have oversimplified the use of the actual traffic measurements. The models above only work if a consistent snapshot of all values can be collected. In practice, this poses significant problems. If the data are not collected for exactly the same time periods, then inconsistencies may occur. There are further problems caused by queues in the routers and bugs in implementing data collection facilities in devices of multiple vendors. The data collection process itself uses unreliable communications (UDP) so that some measurements may be lost due to dropped packets. One approach to overcoming these issues is the use of a separate error correction model, which tries to correct values before feeding them into the models above. Another, shown in Yorke-Smith and Gervet [2002], Yorke-Smith [2004], Yorke-Smith and Gervet [2004] deals with the problem by integrating incomplete and inconsistent data into the constraint solving process.

3.8 Bandwidth Protection

It is very important that a network functions not only when all its elements are working, but that it continues to provide its services when some network elements fail. So far, we have seen two methods of dealing with that issue. In section 3.3, we looked at the provisioning of primary and secondary paths in the network, where the secondary path is used when an element on the primary path fails. Depending on the constraint used, this will protect against single link failures, node failures or SRLG failures. In section 3.7 we have studied the problem of resilience analysis which considers a routed network without an explicit traffic matrix. The resilience analysis provides bounds on the link utilization in a failure event; if the upper bounds are below a given capacity limit, then the service is guaranteed not to be affected after the re-convergence of the routing protocol.

In this section we discuss a different method for bandwidth protection, first described in Xia et al. [2004]. We consider node failures in a network, and try to provide local detours for the traffic around each failed element.

To explain the basic idea, we look at the small example in figure 1 taken from Xia [2005]. In this network we consider the failure of node j , which is currently used to forward traffic from node c to nodes e and f . A set of possible detours will be to use the paths c, k, l, e and c, k, l, f . But how much capacity do we need to allocate on each link of these detours? We don't know the actual values for the flows between c and e or f , but we can come up with bounds on these values from the bandwidth of the links cj , jf and je . The flow between c and f must be less than 20, and the flow between c and e less than 10. But reserving $20 + 10 = 30$ on the detour links ck and kl is wasteful, the combined value of the flows must be below 20, since they both pass through link cj in the normal state of the network. For the link lf we have to allocate 20, and for the link le 10 units of capacity. Indeed, for every link on the detours we have to allocate the maximal amount that could flow through this link, no matter how the flows

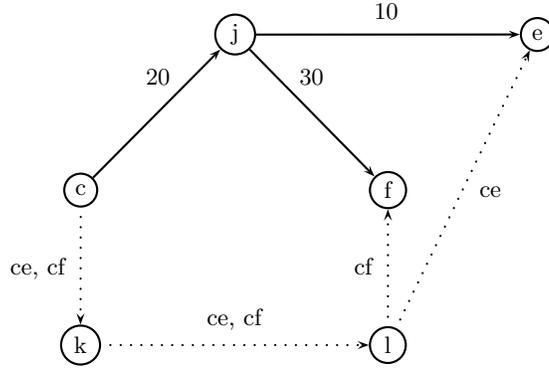


Fig. 1. Bandwidth Protection Problem

through node j are allocated. This leads to the nested optimization problem in (49).

We consider the set \mathbf{F} of all local flows f through the failed element. For each of these flows, we want to provide a single detour around the failure via a link based model from the origin of the flow $\text{orig}(f)$ to its destination $\text{dest}(f)$. For this we introduce $\{0, 1\}$ variables X_{fe} which state whether edge e is used in the detour for flow f . We also use continuous variables Q_{fe} to describe the flow volume for flow f that we need to consider for edge e . These flow quantities are constrained by the capacity of the network around the failed element. For every edge, we have to solve a maximization problem, which finds the flow volume that could be forced through that link in the failure case. A feasible solution consists in a choice of the X_{fe} variables which forms a path from source to destination of flow f and which ensures that on each edge the required capacity is below the edge capacity $\text{cap}(e)$.

$$\begin{array}{l}
 \min_{\{X_{fe}\}} \sum_{f \in \mathbf{F}} \sum_{e \in \mathbf{E}} X_{fe} \\
 \left. \begin{array}{l}
 \forall f \in \mathbf{F} : \\
 \forall e \in \mathbf{E} : \\
 X_{fe} \in \{0, 1\} \\
 \text{quan}(f) \geq Q_{fe} \geq 0
 \end{array} \right\} \text{st.} \left\{ \begin{array}{l}
 \forall n \in \mathbf{N} \setminus \{\text{orig}(f), \text{dest}(f)\} : \\
 n = \text{orig}(f) : \\
 n = \text{dest}(f) : \\
 \max_{\{Q_{fe}\}} \sum_{f \in \mathbf{F}} X_{fe} Q_{fe} \\
 \forall o \in \text{orig}(\mathbf{F}) : \text{ocap}(o) \geq \sum_{f: \text{orig}(f)=o} Q_{fe} \\
 \forall d \in \text{dest}(\mathbf{F}) : \text{dcap}(d) \geq \sum_{f: \text{dest}(f)=d} Q_{fe}
 \end{array} \right. \quad (49)
 \end{array}$$

We have to solve this problem for every failure we want to consider, and then need a mechanism which remembers and activates the detours when a failure occurs. The big advantage of this approach is that it is independent of the traffic pattern at the time of failure. Indeed, we don't need any information about traffic flows, or even the routing method used.

How can we actually solve this type of nested optimization problem? In Xia et al. [2004], two methods are proposed. The first is a transformation into a MILP, using the Karush-Kuhn-Tucker condition Nemhauser and Wolsey [1988] and introducing a set of new variables. This method does not scale well, and can not solve most of the data sets considered in Xia et al. [2004].

The second alternative is to decompose the problem into an integer multi-commodity flow problem Ahuja et al. [1993] as the master problem and capacity optimization problems for each edge as subproblems. The master problem generates tentative assignments for the detours, which are checked by the capacity problems. If the capacity of the edge is exceeded, then a cut is generated which is passed back to the master problem and which will change the path assignment. When no capacity constraints are violated, then a solution is found and the procedure stops. The simplest form of cut that can be generated is a no-good which excludes the current solution, but more effective cuts are possible and are described in Xia et al. [2004].

Experimental results in Xia et al. [2004], Xia [2005] show that while the decomposition method works for some large problem instances, it can be beaten by a different transformation of the embedded optimization problem into MILP model. Scalability of the algorithms is still an issue, as the number of flows to be considered increases with increased connectivity in the graph, which also has an impact on the number of possible detours to be considered.

4 Conclusion

Before we try to summarize the results in this chapter, we want to mention some related problems where constraint programming has been applied. The problem of frequency assignment in radio networks Aardal et al. [2003] has been studied quite successfully with different soft constraint methods, this is discussed in some detail in chapter 9, "Soft Constraints". Constraint programming has also been used for a study of the location of wireless base stations Fruehwirth and Brisset [1998]. An emerging domain with interesting challenges for constraint programming is the area of ad-hoc networks Shang et al. [2003].

In this chapter we have looked at applications for different network problems, considering electrical, water and especially data networks. The applications for data networks cover a wide range of problems, from design, to risk analysis and operational control. Classical finite domain constraint programming currently seems to be rather limited for these problems, this clearly is a field where hybrid systems are achieving much better results. As an explanation we can see two main contributing factors: one is the important role of cost optimization, the other the large scale of the problems together with the fine granularity of the decisions. LP

relaxations and Lagrangian Relaxation (see chapter 15, “Operations Research Methods in Constraint Programming”) seem to provide a much better reasoning on cost bounds than we achieve from individual finite domain or set constraints. At the same time, we often find it easier to construct some feasible solutions and changing them with a restricted neighborhood operator, rather than building very large domain representations from the start.

Limiting factors for the use of hybrid systems are the complexity of designing and evaluating the schemes and the implementation effort required to build a working application. A flexible constraint toolkit like ECLiPSe Cheadle et al. [2003] can help to speed up development, but at the moment building hybrid systems remains still very much a task for specialists.

Is there a way to encapsulate the structure of the problem in some global constraints (see chapter 7, “Global Constraints”), which hide the algorithmic complexity and provide more high-level abstractions for application developers? Global constraints for graph based problems have been around for some time Beldiceanu and Contejean [1994], Bourreau [1999], Bockmayr et al. [2001] and have been very useful for rapid application development in other domains Bourreau [1999], Simonis [1999], Simonis et al. [2000]. As we have seen in section 3.2, they are probably not the right abstraction for the applications discussed here. But there are a number of proposals for new global constraints Beldiceanu et al. [2005], Cambazard and Bourreau [2004], Doms et al. [2005] which may help to solve some of these problems in a more declarative way. Much will depend on how well these constraints will integrate cost reasoning, and which problem size can be handled effectively.

Bibliography

- K. Aardal, S. P. M. van Hoesel, A. M. C. A. Koster, C. Mannino, and A. Sassano. Models and solution techniques for frequency assignment problems. *4OR*, 1 (4):261–317, 2003.
- A. Aggoun and N. Beldiceanu. Extending CHIP in order to solve complex scheduling and placement problems. *Mathematical and Computer Modelling*, pages 57–73, 1993.
- R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows*. Prentice Hall, 1993.
- F. Ajili, R. Rodosek, and A. Eremin. A branch-price-and-propagate approach for optimising IGP weight setting subject to unique shortest paths. In *Proceedings of the 20th Annual ACM Symposium on Applied Computing (ACM SAC '05)*, Santa Fe, New Mexico, March 2005a.
- F. Ajili, R. Rodosek, and A. Eremin. A scalable tabu search algorithm for optimising IGP routing. In *2nd International Network Optimization Conference (INOC '05)*, pages 348–354, March 2005b.
- R. Barták and M. Milano, editors. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Second International Conference, CPAIOR 2005, Prague, Czech Republic, May 30 - June 1, 2005, Proceedings*, volume 3524 of *Lecture Notes in Computer Science*, 2005. Springer. ISBN 3-540-26152-4.
- N. Beldiceanu and E. Contejean. Introducing global constraints in CHIP. *Mathematical and Computer Modelling*, 12:97–123, 1994.
- N. Beldiceanu, P. Flener, and X. Lorca. The tree constraint. In Barták and Milano [2005], pages 64–78. ISBN 3-540-26152-4.
- A. Bockmayr, N. Pisaruk, and A. Aggoun. Network flow problems in constraint programming. In *Principles and Practice of Constraint Programming - CP 2001*, Paphos, Cyprus, November 2001.
- E. Bourreau. *Traitement de Contraintes sur les Graphes en Programmation par Contraintes*. PhD thesis, L'Université de Paris 13 - Institut Galilée Laboratoire d'Informatique de Paris Nord (L.I.P.N.), March 1999.
- M. Brdys, T. Creemers, H. Goosens, J. Riera, A. Heinsbroek, and Z. Lisiak. CLOCWiSe: Constraint logic for operational control of water systems. Technical report, UPC, 2003.
- R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers*, 35(8):677–691, 1986.
- H. Cambazard and E. Bourreau. Conception d'une contrainte globale de chemin. In *JNPC'04*, 2004.
- A. Chabrier. Heuristic branch-and-price-and-cut to solve a network design problem. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems CP-AI-OR 03*, Montreal, Canada, May 2003.

- A. M. Cheadle, W. Harvey, A. J. Sadler, J. Schimpf, K. Shen, and M. G. Wallace. ECLiPSe: An introduction. Technical Report IC-Parc-03-1, IC-Parc, Imperial College London, 2003.
- C. Chiopris and M. Fabris. Optimal management of a large computer network with CHIP. In *2nd Conf Practical Applications of Prolog*, London, UK, April 1994.
- Y. Chu and Q. Xia. Bandwidth-on-demand problem and temporal decomposition. In *2nd International Network Optimization Conference (INOC '05)*, pages 542–550, Lisbon, Portugal, March 2005.
- T. Creemers, L. R. Giralt, J. Riera, C. Ferrarons, J. Rocca, and X. Corbella. Constrained-based maintenance scheduling on an electric power-distribution network. In *3rd Conference on Practical Applications of Prolog (PAP95)*, Paris, France, April 1995.
- T. Creemers, L. Ros, J. Riera, C. Ferrarons, and J. Roca. Smart schedules streamline distribution maintenance. *IEEE Computer Applications in Power*, July 1998.
- W. Cronholm and F. Ajili. Strong cost-based filtering for Lagrange decomposition applied to network design. In M. Wallace, editor, *10th International Conference on Principles and Practice of Constraint Programming (CP 2004)*, pages 726–730, Toronto, Canada, 2004. Springer-Verlag.
- W. Cronholm and F. Ajili. Hybrid branch-and-price for multicast network design. In *2nd International Network Optimization Conference (INOC '05)*, pages 796–802, Lisbon, Portugal, March 2005.
- W. Cronholm, W. Ouaja, and F. Ajili. Strengthening optimality reasoning for a network routing application. In *4th International Workshop on Cooperative Solvers in Constraint Programming (CoSolv '04)*, Toronto, Canada, September 2004.
- W. Cronholm, W. Ouaja, and F. Ajili. Strong reduced cost fixing in network routing. In *2nd International Network Optimization Conference (INOC '05)*, pages 688–694, Lisbon, Portugal, March 2005.
- B. Davie and Y. Rekhter. *MPLS: Technology and Applications*. Morgan Kaufmann Publishers, 2000.
- M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf, and F. Berthier. The constraint logic programming language CHIP. In *FGCS*, pages 693–702, 1988.
- G. Doms, Y. Deville, and P. Dupont. CP(Graph): Introducing a graph computation domain in constraint programming. In van Beek [2005], pages 211–225. ISBN 3-540-29238-1.
- H. El Sakkout and M. Wallace. Probe backtrack search for minimal perturbation in dynamic scheduling. *Constraints*, 5(4):359–388, 2000.
- A. Eremin, F. Ajili, and R. Rodosek. A set-based approach to the optimal IGP weight setting problem. In *2nd International Network Optimization Conference (INOC '05)*, pages 386–392, Lisbon, Portugal, March 2005.
- F. Focacci, A. Lodi, and M. Milano. Cost-based domain filtering. In *Principles and Practice of Constraint Programming - CP 1999*, Alexandria, Virginia, October 1999.

- C. Frei and B. Faltings. Resource allocation in networks using abstraction and constraint satisfaction techniques. In *Principles and Practice of Constraint Programming - CP 1999*, Alexandria, Virginia, October 1999.
- T. Fruehwirth and P. Brisset. Optimal placement of base stations in wireless indoor telecommunication. In *Principles and Practice of Constraint Programming - CP 1998*, Pisa, Italy, October 1998.
- T. Graf, P. Van Hentenryck, C. Pradelles, and L. Zimmer. Simulation of hybrid circuits in constraint logic programming. In *IJCAI-89: Proceedings 11th International Joint Conference on Artificial Intelligence*, pages 72–77, Detroit, 1989.
- T. Hadzic and H. Andersen. Interactive reconfiguration in power supply restoration. In van Beek [2005], pages 767–771. ISBN 3-540-29238-1.
- W. D. Harvey and M. L. Ginsberg. Limited discrepancy search. In *IJCAI (1)*, pages 607–615, 1995.
- N. Heintze, S. Michaylov, and P. Stuckey. CLP(R) and some electrical engineering problems. In J.-L. Lassez, editor, *Logic Programming: Proceedings of 4th International Conference*, pages 675–703, Melbourne, Australia, September 1987. MIT Press.
- J. Jaffar and M. J. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 19/20:503–581, 1994.
- O. Kamarainen. *Local Probing - A New Framework for Combining Local Search with Backtrack Search*. PhD thesis, IC-Parc, Imperial College London, University of London, December 2003.
- O. Kamarainen and H. El Sakkout. Local probing applied to network routing. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems CP-AI-OR 04*, Nice, France, April 2004.
- M. Lauvergne, P. David, and P. Bauzimaault. Connections reservation with rerouting for ATM networks: A hybrid approach with constraints. In P. Van Hentenryck, editor, *Principles and Practice of Constraint Programming - CP 2002*, Cornell University, Ithaca, N.Y., September 2002.
- C. Le Pape, L. Perron, J. Regin, and P. Shaw. Robust and parallel solving of a network design problem. In P. Van Hentenryck, editor, *Principles and Practice of Constraint Programming - CP 2002*, Cornell University, Ithaca, N.Y., September 2002.
- J. Lever. A local search/constraint propagation hybrid for a network routing problem. In *The 17th International FLAIRS Conference (FLAIRS-2004)*, Miami Beach, Florida, May 2004.
- J. Lever. A local search/constraint propagation hybrid for a network routing problem. *International Journal on Artificial Intelligence Tools*, 14(1-2):43–60, 2005.
- V. Liatsos, S. Novello, and H. El Sakkout. A probe backtrack search algorithm for network routing. In *Proceedings of the Third International Workshop on Cooperative Solvers in Constraint Programming, CoSolv'03*, Kinsale, Ireland, September 2003.
- S. Loudni, P. David, and P. Boizumaault. On-line resource allocation for ATM networks with rerouting. In *Integration of AI and OR Techniques in Con-*

- straint Programming for Combinatorial Optimization Problems CP-AI-OR 03*, Montreal, Canada, May 2003.
- R. Malhotra. *IP Routing*. O'Reilly, Sebastopol, CA, 2002.
- K. Marriott and P. Stuckey. *Programming with Constraints: an Introduction*. MIT Press, 1998.
- L. Michel and P. Van Hentenryck. A simple tabu search for warehouse location. *European Journal on Operations Research*, pages 576–591, 2004.
- P. Mirchandani and R. Francis. *Discrete Location Theory*. Wiley, New York, 1990.
- J. T. Moy. *OSPF : Anatomy of an Internet Routing Protocol*. Addison-Wesley, Boston, Ma, 1998.
- G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. Wiley, New York, NY, 1988.
- W. Ouaja and B. Richards. A hybrid solver for optimal routing of bandwidth-guaranteed traffic. In *INOC2003*, pages 441–447, 2003.
- W. Ouaja and B. Richards. A hybrid multicommodity routing algorithm for traffic engineering. *Networks*, 43(3):125–140, 2004.
- W. Ouaja and E. B. Richards. Hybrid Lagrangian relaxation for bandwidth-constrained routing: Knapsack decomposition. In *20th Annual ACM Symposium on Applied Computing (ACM SAC '05)*, pages 383–387, Santa Fe, New Mexico, March 2005.
- L. L. Peterson and B. Davie. *Computer Networks*. Morgan Kaufmann, San Francisco, CA, second edition, 2000.
- S. Prestwich. Three CLP implementations of branch-and-bound optimization. In *Parallelism and Implementation of Logic and Constraint Logic Programming*, volume 2. Nova Science Publishers, Inc, 1999.
- L. Ros, T. Creemers, E. Tourouta, and J. Riera. A global constraint model for integrated routing and scheduling on a transmission network. In *7th International Conference on Information Networks, Systems and Technologies*, Minsk, October 2001.
- Y. Shang, M. P. Fromherz, Y. Zhang, and L. S. Crawford. Constraint-based routing for ad-hoc networks. In *IEEE Int. Conf. on Information Technology: Research and Education (ITRE 2003)*, pages 306–310, Newark, NJ, USA, August 2003.
- H. Simonis. Building industrial applications with constraint programming. In H. Comon, C. Marché, and R. Treinen, editors, *CCL*, volume 2002 of *Lecture Notes in Computer Science*, pages 271–309. Springer, 1999. ISBN 3-540-41950-0.
- H. Simonis. Resilience analysis in MPLS networks. Technical report, Parc Technologies Ltd, 2003.
- H. Simonis. Challenges for constraint programming in networking. In M. Wallace, editor, *Principles and Practice of Constraint Programming - CP 2004*. *10th International Conference*, volume 3258 of *LNCS*, Toronto, Canada, September/October 2004. Springer Verlag.
- H. Simonis and T. Cornelissens. Modelling producer/consumer constraints. In U. Montanari and F. Rossi, editors, *CP*, volume 976 of *Lecture Notes in Computer Science*, pages 449–462. Springer, 1995. ISBN 3-540-60299-2.

- H. Simonis, P. Charlier, and P. Kay. Constraint handling in an integrated transportation problem. *IEEE Intelligent Systems and their applications*, 15(1): 26–32, Jan/Feb 2000.
- B. M. Smith. Symmetry and search in a network design problem. In Barták and Milano [2005], pages 336–350. ISBN 3-540-26152-4.
- J. Symes. Bandwidth-on-demand services using MPLS-TE. In *MPLS World Congress 2004*, Paris, France, February 2004.
- P. van Beek, editor. *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, volume 3709 of *Lecture Notes in Computer Science*, 2005. Springer. ISBN 3-540-29238-1.
- P. Van Hentenryck and J. Carillon. Generality versus specificity: An experience with AI and OR techniques. In *AAAI*, pages 660–664, 1988.
- Q. Xia. Traffic diversion problem: Reformulation and new solutions. In *2nd International Network Optimization Conference (INOC '05)*, pages 235–241, Lisbon, Portugal, March 2005.
- Q. Xia and H. Simonis. Primary/secondary path generation problem: Reformulation, solutions and comparisons. In *4th International Conference on Networking*, Reunion Island, France, 2005. Springer Verlag.
- Q. Xia, A. Eremin, and M. Wallace. Problem decomposition for traffic diversions. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems CP-AI-OR 2004*, pages 348–363, Nice, France, April 2004.
- N. Yorke-Smith. *Reliable Constraint Reasoning with Uncertain Data*. PhD thesis, IC-Parc, Imperial College London, University of London, June 2004.
- N. Yorke-Smith and C. Gervet. On constraint problems with incomplete or erroneous data. In P. Van Hentenryck, editor, *Principles and Practice of Constraint Programming - CP 2002*, Cornell University, Ithaca, N.Y., September 2002.
- N. Yorke-Smith and C. Gervet. Tight and tractable reformulations for uncertain CSPs. In *CP '04 Workshop on Modelling and Reformulating Constraint Satisfaction Problems*, Toronto, Canada, September 2004.