

# Symmetry and Search in a Network Design Problem

Barbara M. Smith

Cork Constraint Computation Centre, University College Cork, Ireland  
b.smith@4c.ucc.ie

**Abstract.** An optimization problem arising in the design of optical fibre networks is discussed. A network contains client nodes, each installed on one or more SONET rings. A constraint programming model of the problem is described and compared with a mixed integer programming formulation. In the CP model the search is decomposed into two stages; first partially solving the problem by deciding how many rings each node should be on, and then making specific assignments of nodes to rings. The model includes implied constraints derived by considering optimal solutions to subproblems. In both the MIP and CP models, it is important to deal with the symmetry of the problem. In the CP model, two sources of symmetry are separated; one is eliminated dynamically during search and the other by assigning ranges rather than explicit values to one set of decision variables. The resulting CP model allows optimal solutions to be found easily for benchmark problems.

## 1 Introduction

In this paper, the development of constraint programming models for an optimization problem arising in the design of optical fibre networks is discussed. The problem was introduced by Sherali, Smith and Lee [8], who discuss the practical scenario that the problem is abstracted from and described mixed integer programming (MIP) formulations. In both the MIP and CP models, it is important to deal with symmetry; Sherali and Smith [7] discuss different ways of dealing with the symmetry in one of the MIP models.

A CP model of a simplified version of the problem (ignoring the demand capacities of the rings), using similar variables and constraints to the MIP model and passed to a CP solver, could solve only small instances. An earlier paper [9] describes how this model evolved into one that could solve larger instances of the full problem; further improvements are presented here. In this paper, Section 2 describes the problem and the MIP model, with the methods in [7] for dealing with symmetry. The following sections describe a CP model for the simplified problem in which the traffic capacity of the rings is ignored; solving this simplified problem is a precursor to solving the full problem. It is useful to separate two sources of symmetry in the problem: Section 7 describes how the symmetry due to the fact that the rings are interchangeable can be eliminated in the CP model. The full problem, taking the traffic capacity into account, is returned to in section 8; this introduces further symmetry from the fact that demand can often be split between rings in different ways. The CP model of the previous sections is adapted, and equivalent solutions are avoided by assigning ranges rather than specific values to the

decision variables. Results are presented showing that optimal solutions for the largest instances in [8, 7] can be found easily. Section 9 draws conclusions.

## 2 Problem Description and MIP Model

Sherali, Smith and Lee [8] describe a network design problem arising from the deployment of synchronous optical networks (SONET). The network contains a number of client nodes and there are known demands (in terms of numbers of channels) between pairs of nodes. (It is explained in [8] that when the model is used for planning, the value assigned to each demand takes account of the uncertainty in predicting the actual demand over the planning horizon.) A SONET ring joins a number of nodes; a node is installed on a ring using an ‘add-drop multiplexer’ (ADM) that is capable of adding and dropping the traffic. Each node can be installed on more than one ring, and traffic can be routed between a pair of client nodes only if they are both installed on the same ring. In this scenario, there is no traffic allowed between rings, but the demand between a pair of nodes can be split between two or more rings. There are capacity limits on the rings (in terms of both nodes and channels). The objective is to minimise the total number of ADMs required, while satisfying all the demands.

The largest test instances used in [8] have 13 nodes, with 24 demand pairs. The rings can accommodate 5 nodes and 40 traffic channels, and there are 7 rings available. (It appears that the cost of SONET rings is negligible, so that there is no practical limit on the number of rings used. The limit is specified in order to model the problem.) 80% of the demand pairs were uniformly generated between 1 and 5, and 20% between 1 and 25. There are also two smaller sets of 15 problems each, one set having 7 nodes and 8 demand pairs and the other 10 nodes and 15 demand pairs.

The data for one of the large instances is given below. The first line gives the origin nodes, the second the destination nodes, and the third the demands in terms of number of channels.

```
1  1  2  2  2  2  2  3  4  4  4  4  5  5  7  7  7  8  8  8  9 10 11 12
9 11  3  5  9 10 13 10 5  8 11 12  6  7  9 10 12 10 12 13 12 13 13 13
8  2 25  5  2  3  4  2  4  1  5  2  5  4  5  2  6  1  4  1  5  9  3  2
```

For this instance, an optimal solution ignoring the traffic levels uses 22 ADMs on 5 rings. The sets of nodes installed on the rings are: {4, 8, 10, 12, 13}, {4, 5, 6, 11}, {1, 2, 9, 11, 13}, {2, 3, 5, 7, 10}, {7, 9, 12}. However, this is not a feasible solution if the level of demand is taken into account, because the fourth ring requires 41 channels. The demand pairs on this ring are not on any other ring, so that the demand cannot be split between two rings. An optimal solution respecting the traffic limit uses 23 ADMs.

The Mixed Integer Programming (MIP) model in [8, 7] is as follows. Let  $N = \{1, \dots, n\}$  be the set of nodes, and  $d_{ij}$  be the traffic demand between nodes  $i$  and  $j$ . This defines a set of edges  $E = \{(i, j), i < j, d_{ij} > 0\}$  and a demand graph  $G(N, E)$ . Let  $M = \{1, \dots, m\}$  be the set of rings,  $r$  be the maximum number of nodes that can be installed on any ring, and  $b$  be the capacity of a ring in terms of the number of channels. Let  $S_i$  be the set of neighbours of  $i$  in  $G$ , i.e. the set of nodes  $j$  such that there

is a demand between nodes  $i$  and  $j$ , so that  $S_i = \{\rho \in E : \rho = (i, j) \text{ or } \rho = (j, i) \text{ for some } j\}$ .

The model has two sets of decision variables:  $x_{ik}, \forall i \in N, k \in M$ , where  $x_{ik} = 1$  if node  $i$  is assigned to ring  $k$ , 0 otherwise;  $f_{\rho k} \forall \rho = (i, j) \in E, k \in M$ , representing the fraction of the demand between the nodes  $i$  and  $j$  that is assigned to ring  $k$ . The model can be stated as:

$$\text{minimize } \sum_{i \in N} \sum_{k \in M} x_{ik}$$

$$\text{subject to: } \sum_{k \in M} f_{\rho k} = 1 \quad \forall \rho \in E \quad (1)$$

$$\sum_{\rho \in E} d_{\rho} f_{\rho k} \leq b \quad \forall k \in M \quad (2)$$

$$\sum_{i \in N} x_{ik} \leq r \quad \forall k \in M \quad (3)$$

$$0 \leq f_{\rho k} \leq x_{ik} \quad \forall i \in N, k \in M, \rho \in S_i \quad (4)$$

$$x_{ik} \in \{0, 1\} \quad \forall i \in N, k \in M \quad (5)$$

This basic model, RD1, has many alternative optimal solutions, because given any feasible network design, equivalent designs can be obtained by “simply reshuffling the demand allocations made to the various individual rings” [7]. Sherali and Smith developed a number of alternative models in which the problem symmetry was reduced by introducing hierarchy into the model. The most successful model, RD3, replaced constraints (3) by:

$$r \geq \sum_{i \in N} x_{i1} \geq \sum_{i \in N} x_{i2} \geq \dots \geq \sum_{i \in N} x_{im}$$

i.e. ring 1 has the largest number of ADMs, followed by ring 2 and so on. In their experiments, this model gave the shortest average run-time, even though it does not eliminate all the symmetry: each ring can accommodate only 5 nodes, and most rings that are used at all will have 4 or 5 nodes allocated to them. The average run-time for the 15 largest instances was just over 21,200 sec., compared with nearly 196,000 sec. for RD1 (on a Sun Ultra 10 Workstation, running CPLEX).

Another model, RD2, which gave an average run-time of over 40,000 sec., replaced constraints (2) by:

$$b \geq \sum_{\rho \in E} d_{\rho} f_{\rho 1} \geq \sum_{\rho \in E} d_{\rho} f_{\rho 2} \dots \geq \sum_{\rho \in E} d_{\rho} f_{\rho m}$$

i.e. ring 1 has the largest demand, followed by ring 2, and so on. Although this model was not as good as the previous model, RD3, by itself, Sherali & Smith found that using the demand allocated to successive rings to break ties in the case that two successive rings have the same number of nodes allocated was promising, and in experiments gave the second smallest average run-time (just under 23,000 sec. on average).

Although even the worst model with symmetry breaking constraints (RD2) did much better than the original model, it is surprising that RD3, using a simple set of constraints which leave much of the symmetry intact, did so well. Sherali and Smith concluded that this was because the constraints added tended to encourage integral solutions, since the nonzero coefficients are all 1s. In CP, on the other hand, experience is that as much as possible of the symmetry in the problem should be eliminated. In CP models of the SONENT problem, as shown later, all the symmetry can be eliminated quite easily and this contributes to its successful solution.

### 3 The Unlimited Traffic Capacity Problem

This section describes a CP model for a simplified version of the problem in which the demand capacity of the rings is ignored. The problem initially proved difficult to solve using CP, so that it was helpful to start with a simplified problem. Moreover, solving the simpler problem gives a good lower bound on the solution to the full problem, in fact often a feasible solution, so that it is useful to solve the simpler problem even now that better models have been developed. Finally, considering the simpler problem and the full problem separately makes explicit the two sources of symmetry in the problem, since one arises only in the full problem. These sources of symmetry can be eliminated in different ways. Solution of the full problem will be described in section 8.

For the simplified problem, the relevant data can be represented by the demand graph,  $G(N, E)$ . Figure 1 gives an example, representing one of the large instances.

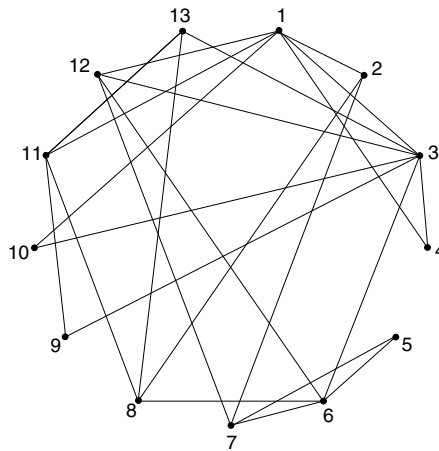


Fig. 1. Demand pairs in a sample instance of the SONENT problem

It is possible to build a CP model the unlimited traffic capacity problem using only the  $x_{ik}$  variables of the MIP model. However, it was found that the resulting model could only be solved in a reasonable time (using ILOG Solver) for the 15 smallest instances. In any case, the expression of the constraint that if there is a demand between a pair of nodes they must both appear on the same ring is somewhat awkward for CP.

Partly to make it easier to express this constraint, two dual sets of set variables were introduced: variable  $R_k$  represents the set of nodes assigned to ring  $k$ ,  $\forall k \in M$ , and  $N_i$  is the set of rings that node  $i$  is assigned to,  $\forall i \in N$ .

The model can then be written as:

$$\text{minimize } t = \sum_{i \in N} \sum_{k \in M} x_{ik} = \sum_k |R_k| = \sum_i |N_i|$$

$$\text{subject to: } |N_i \cap N_j| \geq 1 \quad \forall (i, j) \in E \quad (6)$$

$$|R_k| \leq r \quad \forall k \in M \quad (7)$$

$$(x_{ik} = 1) = (i \in R_k) = (k \in N_i) \quad \forall i \in N, k \in M \quad (8)$$

$$x_{ik} \in \{0, 1\} \quad \forall i \in N, k \in M \quad (9)$$

The first constraint expresses that if there is a demand between nodes  $i$  and  $j$ , there must be at least one ring that they are both assigned to. The second could alternatively be expressed as:  $\sum_i x_{ik} \leq r$ , and so is equivalent to (3) in the MIP model. The third constraint is a set of channelling constraints linking the old and new variables.

The  $x_{ik}$  variables are used as the search variables and an optimal solution is found using the branch-and-bound optimization provided in ILOG Solver: whenever a solution is found, Solver adds a constraint that in future solutions, the value of  $t$  (the objective variable) must be smaller. Hence, when there is no solution satisfying the current constraint on  $t$ , the last solution found is known to be optimal.

## 4 How Many ADMs for Each Node?

The approach described in the last section can solve the small and medium SONET instances to a limited extent. With suitable variable ordering heuristics, good, and sometimes optimal, solutions for these instances can be found very quickly, but proving optimality is slow; as before, of course, we are still dealing only with the unlimited traffic capacity problem.

Part of the difficulty is that very little can be deduced from the assignment of a value to one of the  $x_{ik}$  variables, or even from several such assignments. In particular, when it comes to proving optimality, it is hard to tell from such assignments whether the current bound on the objective can be met. Thus, the search explores many unprofitable branches, with no means of pruning them.

As described in [9], the proof of optimality can be speeded up by focusing on the number of ADMs each node needs. Hence, for each node, an integer variable  $n_i$  is introduced, where  $n_i = |N_i|$ , i.e.  $n_i$  is the number of rings that node  $i$  is installed on.

Deciding how many rings each node should appear on is not sufficient to solve the problem, so the  $n_i$  variables cannot be used alone as search variables. After the  $n_i$  variables have been assigned, the search continues by assigning values to the  $x_{ik}$  variables. Once it has been decided how many rings each node should appear on, it is easy to find a consistent assignment of nodes to rings, or prove that there is no such assignment. This is still a complete search, since if no assignment to the  $x_{ik}$  variables can be found, the search backtracks to the  $n_i$  variables.

Hence, the problem is decomposed into first deciding how many rings each node should be on, and then assigning each node to a specific set of rings. A similar approach is common in scheduling and has proved useful for other problems, for instance in [4].

A further improvement comes from assigning increasing values of the objective variable until a solution is found, rather than using the branch-and-bound procedure described earlier. This is easily implemented by making  $t$  a search variable and assigning it first. One benefit is that, since the objective  $t$  is equal to the sum of the  $n_i$  variables, fixing  $t$  reduces their domains. Once again, if no feasible assignment to the  $n_i$  variables can be found, the search backtracks to try a larger value of  $t$ ; hence, the first solution found must be optimal.

This would not be a useful solution strategy if finding the optimal solution were very difficult and we were prepared to settle for a good solution and forgo the proof of optimality. Alternatively, if there were a large gap between the smallest value of  $t$  found by propagating the constraints and the optimal value, proving that every intervening value is infeasible might take too long. For these instances of the SONET problem, the minimum value of  $t$  which need be considered is between 18 and 20 (given the implied constraints on the  $n_i$  variables described in the next section) and the optimal value is between 20 and 24, so that there are only a few infeasible values of  $t$  to consider before the optimal value is reached.

## 5 Implied Constraints

It has proved much easier to reason about the number of rings that each node must be installed on than whether or not node  $i$  should be installed on ring  $k$ . Consequently, the  $n_i$  variables have been a fruitful source of implied constraints, allowing infeasible assignments to be detected early.

The simplest constraint is that if the degree  $\delta_i$  of node  $i \geq r$ , it must be placed on more than one ring: for instance, in Figure 1, nodes 1, 3 and 6 must each appear on at least 2 rings. In general,  $n_i \geq \lceil \frac{\delta_i}{r-1} \rceil$ .

A similar constraint was used in [8], but only for the node with largest degree in the demand graph: this node is assigned to the first  $n_i$  rings in order to reduce the symmetry in the problem. Here, we do not need such a constraint to deal with symmetry (as discussed below in section 7), but the set of constraints is useful in detecting when the current value of the objective cannot be attained.

The remaining implied constraints have been derived by considering subproblems consisting of pairs of nodes with a demand between them, and their neighbours in the demand graph.

- if two nodes  $i$  and  $j$  are connected, and each of them is connected to fewer than  $r$  other nodes (so that, considered individually, it appears that each could be placed on just one ring), but together they are connected to at least  $r - 1$  other nodes, then at least one of them must be on at least two rings i.e.  $n_i + n_j \geq 3$ .

In Figure 1, nodes 9 and 11 are connected to nodes 1, 3, 8 and 13, and to each other. There must be a ring that both nodes are installed on, since they are connected, and both sets of neighbours cannot fit onto this ring as well. Hence, one of nodes 9 and 11 must be installed on another ring.

- if two nodes  $i$  and  $j$  are connected, and one is connected to at least  $r$  nodes (so that the last constraint does not apply), but the other is not, and they are connected to a total of more than  $2r-3$  other nodes between them, then nodes  $i$  and  $j$  require at least four ADMs, i.e.  $n_i + n_j \geq 4$ .

For instance, for the example of Figure 1, node 3 must be on at least 2 rings, whereas node 13 could be installed on a single ring, with its neighbours. One more of node 3's neighbours could be on the same ring, but this would leave 5 of the demand pairs involving node 3 not yet accommodated, and this would require at least 2 more rings. Hence  $n_3 + n_{13} \geq 4$ .

- Suppose that two nodes  $i, j$  are not connected; that each could be installed on just one ring but they could not both be installed on the same ring (i.e.  $\delta_i \leq r-1$ ,  $\delta_j \leq r-1$ ,  $\delta_i + \delta_j \geq r-1$ ); and they have a mutual neighbour  $k$  which has more than  $r-1$  neighbours. (In Figure 1, nodes 2, 11 and 1 meet these conditions.) If nodes  $i$  and  $j$  appear on just one ring each, then node  $k$  must also be installed on these rings; if the three nodes have more than  $2r-4$  other distinct neighbours in total, then node  $k$  must also appear on a third ring. We can add a constraint to reflect this: if  $n_i = 1$  and  $n_j = 1$  then  $n_k \geq 3$ .

For instance, nodes 2, 11 and 1 have more than  $2r-4$  neighbours (nodes 3, 4, 7, 8, 9, 10, 13). The ring with nodes 2 and 1 could also accommodate two more of node 1's neighbours; the ring with nodes 11 and 1 could accommodate another; but this still leaves one of the demand pairs involving node 1 unplaced, requiring another ring. Hence, if  $n_2 = 1$  and  $n_{11} = 1$  then  $n_1 \geq 3$ .

In [8], valid inequalities are derived which similarly express a lower bound on the number of ADMs required by a subset of the nodes. The inequalities are generated by selecting subsets of at least  $r$  nodes (where  $r$  is the maximum number of nodes on any ring), or a set of nodes with total demand between them greater than  $b$  (where  $b$  is the maximum demand capacity of a ring). For instance, any connected set of  $r+1$  nodes requires at least  $r+2$  ADMs. In [8], these inequalities are reported to reduce runtime by about one-third, with some care in selecting appropriate sets of nodes. However, in the CP model, they would be weaker than the implied constraints listed earlier.

The foregoing implied constraints would still be valid in the full problem, i.e. taking into account the traffic capacity of the rings. The following constraint would not then be valid, but is useful in the simplified problem:

- the number of rings that a node is installed on should be no more than its degree, i.e.  $n_i \leq \delta_i$ .

The following dominance rules can also be added. They are not true of every consistent solution, and so are not logical consequences of the existing constraints, as implied constraints are, but they will be satisfied by at least one optimal solution:

- a ring cannot have just one node on it, i.e.  $|R_k| \neq 1$ , for  $1 \leq k \leq m$ . In fact, any ring must have two *connected* nodes on it.
- the total number of nodes allocated to two non-empty rings must be more than the number that can be accommodated on one ring, i.e. if  $|R_k| > 0$  and  $|R_l| > 0$  then  $|R_k| + |R_l| > r$ , for all  $k, l$  with  $1 \leq k < l \leq m$ . Otherwise there is an equally good solution in which the two rings are combined into one.

All these additional implied constraints and dominance rules are useful: they reduce both search and runtime.

Once a complete assignment to the  $n_i$  variables has been found, consistent with all these constraints, the following constraints are useful while the  $x_{ik}$  variables are being assigned. They help to complete the solution quickly or decide that it cannot be completed.

- if  $n_i = 1$ , and node  $i$  is installed on ring  $k$ , then all the neighbours of node  $i$  must also be on ring  $k$ , i.e. if  $S_i$  is the set of neighbours of node  $i$ :

$$\text{if } n_i = 1 \text{ and } x_{ik} = 1 \text{ then } \{i\} \cup S_i \subseteq R_k \quad \text{for } 1 \leq i \leq n, 1 \leq k \leq m$$

- similarly, if  $n_i = 2$ , once node  $i$  has been allocated to two rings, all its neighbours must be on these two rings as well, i.e.

$$\begin{aligned} \text{if } n_i = 2 \text{ and } x_{ik} = 1 \text{ and } x_{il} = 1 \text{ then } \{i\} \cup S_i \subseteq R_k \cup R_l \\ \text{for } 1 \leq i \leq n, 1 \leq k < l \leq m \end{aligned}$$

## 6 Variable Ordering Heuristics

The search strategy described earlier requires two variable ordering heuristics, one for the  $n_i$  variables and one for the  $x_{ik}$  variables. The heuristics described in this section are used both for the simplified problem and the full problem that includes the traffic capacities of the rings.

Devising variable ordering heuristics for the first stage of search i.e. for assigning the  $n_i$  variables, is more complicated than when there is only one set of search variables. The aim is not just to find an assignment to satisfy the direct constraints on the first-stage search variables (which is easily done), but to find one that can be extended to the second-stage variables. This makes it difficult to predict the behaviour of a proposed heuristic.

Three variable ordering heuristics have been compared: smallest domain, minimum degree and maximum degree. (The last two are static orders, using the original degrees in the demand graph.) When the objective variable is assigned first, any variable ordering will generate the same set of sub-optimal complete assignments to the  $n_i$  variables (i.e. assignments for which the value of  $t$  is less than its optimal value). This is because all sub-optimal assignments satisfying the implied constraints on the  $n_i$  variables must be found, whatever heuristic is used, and will fail only when the second stage tries to assign the  $x_{ik}$  variables. The different heuristics tested sometimes require slightly different numbers of backtracks to explore the suboptimal assignments; however, they principally differ in the number of complete assignments to the  $n_i$  variables that they consider at the optimal value of  $t$ , before finding one that leads to a solution. On average, minimum degree is the best of the three heuristics, with smallest domain slightly worse, and maximum degree significantly worse. Clearly, the total number of complete assignments to the  $n_i$  variables at the optimal value of  $t$  is the same for all three heuristics, but they each consider them in a different order, and hence find a solution earlier

or later. Poor performance of a heuristic indicates that the assignments it considers first are those that are less likely to be successfully extended to the  $x_{ik}$  variables, but it is difficult to identify the reason.

A number of variable ordering heuristics have been investigated for the  $x_{ik}$  variables, which are assigned once a complete assignment to the  $n_i$  variables has been found. In all cases, the smallest numbered ring not yet fully occupied is chosen, and then a node chosen to place on this ring. The value ordering is to choose 1 before 0, i.e. when considering variable  $x_{ik}$ , choose to place node  $i$  on ring  $k$  before choosing not to place it. The best heuristic found chooses the node  $i$  assigned to fewest rings (i.e. for which  $n_i$  is smallest), breaking ties by choosing the node with largest degree. This means that any node that is only on one ring (i.e.  $n_i = 1$ ) is placed first. The constraint in section 5, that if  $n_i = 1$  and  $x_{ik} = 1$ , then every neighbour of node  $i$  must be on ring  $k$ , will then be triggered. By choosing the node with maximum degree, the largest number of neighbouring nodes will be placed as a result of this constraint. Since all these nodes have to be on the same ring as node  $i$ , given the first-stage assignment, postponing placing them is likely to lead to future failure and wasted search.

The heuristics investigated were selected largely by trial and error; although those chosen have been found to be superior to the others considered, it is unlikely that they are the best possible heuristics for these models. Nevertheless, the selected heuristics are giving good results.

## 7 Symmetry Breaking

Sherali & Smith's aim in [7] was to investigate ways of dealing with symmetry in MIP models in order to speed up solution time: the SONET problem was one of several considered. Symmetry also causes difficulties for search in constraint programming. The symmetry in the SONET problem partly arises because the available rings are indistinguishable: a solution is unchanged by permuting the rings, with their associated nodes. This symmetry could be eliminated in a similar way to [7] by adding constraints to the model to distinguish them; for instance, constraints that the set of nodes installed on ring  $k$  is lexicographically greater than the set of nodes on ring  $k+1$  (so that any empty rings are the largest numbered). However, it is important to ensure that symmetry-breaking constraints do not conflict with the variable ordering. If they do, it can happen that the solutions in a symmetry equivalence class which the constraints eliminate are those that would be found earlier, given the variable and value ordering, than the solutions allowed by the constraints. Thus, finding a solution can be hindered rather than helped by the symmetry breaking. As discussed earlier, a dynamic variable ordering heuristic is used when assigning nodes to rings, and this makes it difficult to ensure that symmetry-breaking constraints are consistent with the variable ordering.

An alternative approach is to eliminate the symmetry using SBDS (Symmetry Breaking During Search) [5]. This adds constraints during search, on backtracking to a choice point, to ensure that no partial assignments symmetrically equivalent to those already considered will be considered in future. Since the constraints are added dynamically, depending on the choices already made, SBDS is compatible with any variable ordering, including dynamic orderings. SBDS requires the user to supply a function describing

the effect of each symmetry on the assignment of a value to a variable. It is particularly simple to use when the symmetry is equivalent to permuting the rows or columns of a matrix of variables, since in that case the symmetry functions need only describe the transpositions of pairs of rows/columns, not all their permutations. Here, if the variables  $x_{ik}$  are thought of as corresponding to the elements of a 2-dimensional matrix, the columns of the matrix (corresponding to the 2nd subscript) can be permuted and hence, SBDS only requires functions describing the effects of transposing pairs of rings.

The symmetry breaking functions required by SBDS were first written for the original CP model, using only the  $x_{ik}$  variables. With the two-stage search described in section 4, the symmetry between the rings can be broken in the same way, using the same SBDS functions. Symmetry breaking is only needed when assigning values to the  $x_{ik}$  variables; the symmetry does not affect the  $n_i$  variables. This demonstrates once again the usefulness of having a symmetry breaking method that does not depend on the search order: with the two-stage search, it would be even more difficult than before to use symmetry-breaking constraints and ensure compatibility with the variable ordering, since the order in which the  $x_{ik}$  variables are assigned depends on the previous assignments to the  $n_i$  variables, as described in section 6.

The effect of using SBDS on the effort required to solve the SONET problem (ignoring the traffic capacity) is shown in Table 1. The model used includes the search strategy and variable ordering heuristics described earlier. The 15 medium-sized instances (10 nodes, 15 demand pairs, 6 available rings) are trivial if the symmetry is eliminated; if not, some of them take much longer to solve. For the larger instances, with 13 nodes,

**Table 1.** Solving medium-sized SONET instances, with and without symmetry-breaking, using ILOG Solver 6.0. ‘Value’ is the minimum number of ADMs required. ‘Backtracks’ is the number of times the search backtracks following failure, reported by Solver. ‘Time’ is the cpu time in seconds on a 1.7GHz Pentium M PC

Instance	Value	With SBDS		No symmetry breaking	
		Backtracks	Time	Backtracks	Time
1	14	14	0.03	596	0.21
2	14	11	0.02	27	0.03
3	14	25	0.03	292	0.12
4	13	43	0.04	917	0.31
5	15	49	0.04	7,423	2.35
6	14	19	0.02	20	0.03
7	13	9	0.03	17	0.02
8	14	12	0.02	13	0.02
9	15	33	0.03	499	0.18
10	14	50	0.05	2,270	0.70
11	12	1	0.02	9	0.03
12	15	50	0.03	776	0.27
13	15	104	0.05	2,361	0.74
14	15	270	0.18	28,960	9.14
15	15	28	0.02	667	0.21
Average		47.86	0.04	2,989.13	0.96

24 demand pairs and 7 available rings, it is essential to break the symmetry in order to solve them in a reasonable time: being larger, these problems are more difficult to solve even when the symmetry is eliminated, and the larger number of available rings means that the number of symmetrically equivalent solutions is up to  $7!$ , rather than  $6!$ . (Given  $m$  available rings, there are fewer than  $m!$  symmetrically equivalent solutions, since swapping a pair of empty rings does not change the solution.)

## 8 Modelling Traffic Capacity and Demand Splitting

As stated earlier, in solving the problem ignoring the traffic capacity we can at least get a lower bound on the number of ADMs required in the full problem. In fact, several of the solutions found for the large instances are still feasible, and therefore optimal, when the demand levels between pairs of nodes and the traffic capacities of the rings are added. Hence, developing a model to solve the simpler problem is not simply a stage on the way to solving the full problem: the solutions found can be useful.

As in the MIP model, the demand capacity of the rings can be modelled by introducing variables  $f_{\rho k}$  for  $\rho = (i, j) \in E$  and  $k \in M$ .  $f_{\rho k}$  is the fraction of the demand between nodes  $i$  and  $j$  that is assigned to ring  $k$ , and  $0 \leq f_{\rho k} \leq 1$ .

The constraints of these variables from the MIP model can be used without change. In the CP model, most  $f_{\rho k}$  variables are set to 0 or 1 by these constraints as the  $x_{ik}$  variables are assigned. If neither node in a demand pair, corresponding to  $\rho \in E$ , appears on ring  $k$ , the corresponding  $f_{\rho k}$  variable will be set to 0; if a demand pair appears on only one ring, the fraction of the demand on that ring is 1. Relatively few demand pairs appear on more than one ring, and so require reasoning about the fraction of the demand allocated to each.

In the MIP models, the variables  $f_{\rho k}$  are decision variables. However, this creates another source of alternative equivalent solutions, since there are usually several ways of splitting the demand between rings, given a feasible allocation of nodes to rings. For instance, the optimal solution for the example in section 2, taking the demand levels into account, uses 23 ADMs. The sets of nodes on each ring are:  $\{4,8,10,12,13\}$ ,  $\{1,4,9,11,13\}$ ,  $\{4,5,6\}$ ,  $\{2,3,7,10,13\}$ ,  $\{2,5,7,9,12\}$ . The total traffic demand between the nodes on the fourth ring is 45 channels, whereas the capacity of a ring is 40. However, this solution is feasible, because nodes 10 and 13, with a demand of 9 units between them, are also both installed on the first ring, which has enough spare capacity to take all the demand between these two nodes. Hence, a feasible solution can allocate 0, 1, 2, 3 or 4 channels on the first ring to the demand between nodes 10 and 13 and the remaining demand from this pair to the fourth ring.

Some of the formulations in [7] attempt to reduce the effects of this form of symmetry. For instance, RD2, discussed in section 2, has constraints to ensure that the total demand on successive rings is non-increasing, and these constraints reduce both forms of symmetry in the problem, to some extent. However, this model does not give as good performance as RD3, which only reduces the symmetry between the rings.

Constraint programming offers a different way to avoid considering equivalent alternative solutions in this case; it is not necessary to decide how the demand between a

pair of nodes is to be split between rings, but simply to ensure that it can be done feasibly. This can be done in ILOG Solver: the bounds on the  $f_{\rho k}$  variables can be reduced until they are consistent with the constraints acting on them. For instance, in the example given earlier where the nine channels required between nodes 10 and 13 can be split between two rings, with up to four channels on one ring and the remainder on the other, the domain of the relevant variables would be reduced to  $[0, \dots, 0.4444]$  and  $[0.5555, \dots, 1]$ . To take another simple case, if two nodes forming a demand pair are installed on two rings, both of which can accommodate all the demand between the pair, the ranges of the demand fraction variables will both be left as  $[0, \dots, 1]$ , indicating that any fraction of the demand can be accommodated on one ring and the rest on the other. In this way, the alternative solutions from splitting the demand in different ways are not explicitly considered. When the bounds of the variables have been reduced, any value remaining (that gives an integral number of channels as a fraction of the total demand between a pair of nodes) can form part of a feasible solution consistent with the traffic capacity of the rings. It is straightforward then to construct a solution by choosing say the smallest value in some variable's domain (which must be one of the allowed fractions), re-establishing consistency and continuing in this fashion until every variable has been assigned.

A few other minor changes to the previous model are required to deal with the traffic capacity. The constraint that any two rings must have a total of more than  $r$  nodes installed on them is no longer correct, but can be modified so that two rings *either* have more than  $r$  nodes *or* more than  $b$  channels between them. Further, it is no longer necessarily true that a node with degree  $\delta_i$  should appear on at most  $\delta_i$  rings, and this constraint is dropped.

**Table 2.** Solving the SONET problem for the large instances, either ignoring the traffic capacity of the rings, or taking it into account. 'Optimum' is the minimum number of ADMs in each case

Instance	Without traffic capacity			With traffic capacity		
	Backtracks	Time	Optimum	Backtracks	Time	Optimum
1	543	0.41	22	990	0.95	22
2	445	0.42	20	451	0.65	20
3	401	0.42	22	417	0.62	22
4	680	0.55	23	1,419	1.52	23
5	150	0.07	20	922	0.70	22
6	948	0.85	22	306	0.29	22
7	86	0.11	20	982	1.15	22
8	40	0.07	20	34	0.09	20
9	2,280	2.11	22	35,359	45.13	23
10	2,280	1.77	23	4,620	6.75	24
11	349	0.35	22	352	0.54	22
12	48	0.05	20	1,038	1.09	22
13	88	0.09	21	105	0.14	21
14	558	0.45	23	1,487	1.66	23
15	1,038	0.97	22	13,662	19.59	23
Average	662.27	0.58		4142.93	5.39	

Table 2 shows the results with this strategy. When taking the traffic capacity into account, the revised model was used to find a new optimal solution from scratch in all cases, even when the solution already found was still feasible and therefore optimal. The average solution time for the instances used in the Sherali & Smith paper is now only a few seconds. The average is dominated by just two of the problems (9 and 15) for which satisfying the traffic capacity requires a larger number of ADMs than the simpler problem; further improvements to the CP model will focus on problems of this kind. In [7], the average solution time (with the level of symmetry breaking giving best performance) was 21,220 sec., although on Sun Ultra 10 (a much slower machine), and the corresponding average number of nodes in the branch and bound tree using CPLEX was about 600. It is hard to compare the performance of the CP model and the MIP model, but it is clear that these instances are now easy to solve using the CP model and search strategy described.

## 9 Conclusions

Sherali and Smith [7] investigated different possible ways of reducing the symmetry in the MIP model of the SONET problem. The most successful reformulation of those that they compared (RD3) leaves much of the symmetry intact: it imposes constraints that the number of nodes installed on successive rings is non-increasing. Optimal solutions for their sample instances have several rings with the same number of nodes, which will not be distinguished by these constraints. Furthermore, the RD3 mode does not address the symmetry due to the fact that when a demand is split between two or more rings it can often be done in different ways. In the CP model, the symmetry due to the rings being interchangeable can be eliminated using SBDS: this can reduce the run-time to solve the simplified problem for the medium-sized instances by more than an order of magnitude, as shown in Table 1. In view of the importance of eliminating the symmetry in the CP model, it is surprising that in the MIP approach reducing the symmetry, rather than eliminating it completely, is sufficient to achieve the best results. It is not clear what the implications are for hybrid MIP/CP algorithms for problems with symmetry.

The remaining symmetry in the problem, from splitting the demand in different ways, can also easily be eliminated in the CP model by finding feasible ranges of values for the fraction of demand on each ring, rather than choosing specific values. For a demand fraction variable,  $f_{\rho k}$ , whose domain is still an interval after the domains are made consistent with the constraints, the values in the interval that correspond to integral numbers of channels can be seen as *partially interchangeable*, as defined by Choueiry and Noubir [2]. In their definition, two values for a (discrete-valued) variable in a constraint satisfaction problem are partially interchangeable with respect to a subset  $A$  of variables if and only if any solution involving one value implies a solution involving the other, with possibly different values for variables in  $A$ . The set  $A$  for a variable  $f_{\rho k}$  consists of the variables  $f_{\rho k'}$  for  $k' \in M$ ,  $k' \neq k$ , whose domain is an interval rather than a fixed value after its bounds have been reduced, i.e.  $A$  consists of the fraction variables for the same demand pair on other rings whose values are neither 0 nor 1. Choueiry and Noubir give an algorithm to iden-

tify and exploit partially interchangeable values, but in this case, it is not required: the method described in section 8 is sufficient. This demonstrates that partially interchangeable values can in some problems be easily exploited to allow the possible alternative solutions to be implicitly found, without enumerating them or choosing between them.

The CP model explicitly represents more features of the problem than the MIP models (the set of nodes on each ring, the set of rings each node is on, the number of rings each node is on), using multiple sets of variables, linked by channelling constraints. This richer model allows the decomposition of the search into two stages: first deciding how many rings each node should be on, and then assigning the nodes to specific rings, backtracking to the first stage if the assignment in the second stage fails. Although a similar decomposition has been used in other problems, it is worth noting that much previous work in CP using redundant models linked by channelling constraints [1] has used one model as a primal model, providing the search variables. The CP model of the SONET problem is an example of the more complex possibilities in using redundant models.

Modelling the problem in this way also allows implied constraints to be derived on the variables representing the number of rings each node is on. The implied constraints have been derived by hand from considering subgraphs of the demand graph, consisting of pairs of connected nodes and their neighbours, and finding lower bounds on the number of rings the pair of nodes must be installed on. It would be possible to extend these constraints to take account of the level of demand between the nodes, and not just the existence of a demand. This was done, for instance, in deriving valid inequalities for the MIP model [8]. It would also be possible to derive further implied constraints systematically by solving subproblems consisting of two or three connected nodes and their neighbours in the demand graph. The implied constraints already used have proved invaluable, and such a systematic approach, taking the demand levels into account, could lead to further improvements. Generating implied constraints from solutions to subproblems could also be a useful approach in CP models for other problems.

Considerable effort has been put into developing the CP model from the initial simple model based on the MIP model. Remodelling currently requires expertise in constraint programming, and it would be preferable if the process could be automated or at least supported. A proposal for a system to do this is presented by Frisch *et al.* [3], using models of the SONET problem as an illustration.

The SONET problem initially appeared intractable for constraint programming; remodelling has eventually resulted in a CP model that appears competitive with Sherali & Smith's MIP model. The largest instances that they solved can be solved with the CP model in an average of 30 sec. on a 600 MHz Celeron PC. Optimization is generally considered to be difficult for CP, but this experience suggests that successful CP models could be developed for other initially unpromising optimization problems too. Régis [6], for instance, has developed a CP approach to solving the maximum clique problem and has shown that it is competitive with existing methods, achieving new solutions to benchmark problems. Improved CP models might improve overall performance even for problems that are eventually solved using a CP/IP hybrid.

## Acknowledgments

I should like to thank other members of the APES and CPPod groups (<http://www.dcs.st-and.ac.uk/~apes>, <http://www.dcs.st-and.ac.uk/~cppod>), especially Ian Miguel, for their encouragement and interest through several iterations of work on this problem. I am also grateful to Hanif Sherali and Cole Smith for sending me their sample instances, and to Sarah Fores and Les Proll for bringing the problem to my attention. This material is based in part on work done while the author was employed at the University of Huddersfield, U.K., and in part on works supported by the Science Foundation Ireland under Grant No. 00/PI.1/C075.

## References

1. B. M. W. Cheng, K. M. F. Choi, J. H. M. Lee, and J. C. K. Wu. Increasing constraint propagation by redundant modeling: an experience report. *Constraints*, 4:167–192, 1999.
2. B. Y. Choueiry and G. Noubir. On the Computation of Local Interchangeability in Discrete Constraint Satisfaction Problems. In *AAAI-98*, pages 326–333, 1998.
3. A. M. Frisch, B. Hnich, I. Miguel, B. M. Smith, and T. Walsh. Towards Model Reformulation at Multiple Levels of Abstraction. In A. M. Frisch, editor, *Proceedings of the International Workshop on Reformulating Constraint Satisfaction Problems: Towards Systematisation and Automation*, 2002.
4. A. M. Frisch, I. Miguel, and T. Walsh. Modelling a Steel Mill Slab Design Problem. In *Proceedings of the IJCAI'01 Workshop on Modelling and Solving Problems with Constraints*, pages 39–45, 2001.
5. I. P. Gent and B. M. Smith. Symmetry Breaking During Search in Constraint Programming. In W. Horn, editor, *Proceedings ECAI'2000*, pages 599–603, 2000.
6. J.-C. Régin. Using Constraint Programming to Solve the Maximum Clique Problem. In F. Rossi, editor, *Principles and Practice of Constraint Programming - CP 2003*, LNCS 2833, pages 634–648. Springer, 2003.
7. H. D. Sherali and J. C. Smith. Improving Discrete Model Representations Via Symmetry Considerations. *Management Science*, 47:1396–1407, 2001.
8. H. D. Sherali, J. C. Smith, and Y. Lee. A branch-and-cut algorithm for solving an intra-ring synchronous optical network design problem allowing demand splitting. *INFORMS J. on Computing*, 12:284–298, 2000.
9. B. M. Smith. Search Strategies for Optimization: Modelling the SONET Problem. Report APES-70-2003, Aug. 2003. Presented at the CP'03 Workshop on Modelling and Reformulating Constraint Satisfaction Problems.