

Search Strategies for Optimization: Modelling the SONET Problem

Report APES-70-2003, August 2003.

Available online from <http://www.dcs.st-and.ac.uk/~apes>

Barbara M. Smith

School of Computing & Engineering, University of Huddersfield, U.K.

`b.m.smith@hud.ac.uk`

Abstract. An optimization problem arising in the design of optical fibre networks is considered. Initially, it proved very difficult to solve using constraint programming: a straightforward CP model could only solve very small instances. It was remodelled using a variety of CP modelling techniques that are by now standard: symmetry breaking, implied constraints, new variables linked to the original search variables by channelling constraints. A novel two-phase search has been introduced; first assigning values to a set of intermediate variables in order to guide the subsequent assignments to the original search variables. Variable ordering heuristics have a huge impact on search effort; the two-phase search complicates the choice, but good heuristics have been found which allow the test instances to be solved. Finally, the optimization process has been changed to guide the search better.

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 studied by Sherali and Smith [6] who used a mixed integer linear programming model. A model using similar variables and constraints, translated into a CP solver, could solve only very small instances of a simplified version of the problem.

The problem has been remodelled, using the armoury of CP modelling, including symmetry breaking; introduction of implied constraints (including both those logically implied by the other constraints, and those implied by the fact that this is an optimization problem); new variables representing different aspects of the problem linked to the original variables by channelling constraints. The most important feature of the remodelling has been the design of the search strategy: firstly, the search for solutions is conducted in two phases, first assigning values to a set of intermediate variables, insufficient by themselves to solve the problem, and then to the original search variables. Secondly, the variable ordering heuristics used are of crucial importance; several choices are discussed. Finally, the usual approach of using successive values of the objective to provide a bound on future solutions has been changed to avoid wasted effort considering non-optimal solutions.

2 Problem Description

The problem is described by Sherali and Smith [6] in a paper on dealing with symmetry in linear programming problems (integer and mixed integer) by adding symmetry-breaking constraints to the model. They treat three different problems including the SONET problem.

To quote from Sherali & Smith: “The SONET is a standard of transmission technology over optical fiber networks.” The network contains a number of client nodes and there are known demands (in terms of numbers of channels) between pairs of nodes. A SONET ring joins a number of nodes; a node is installed on a SONET 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: there is no traffic allowed between 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 15 randomly-generated test instances used by Sherali & Smith have 13 nodes, with 24 demand pairs. The SONET rings can accommodate 5 nodes and 40 traffic channels, and there are 7 rings available. (From other descriptions, 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 each¹.

A sample set of demand pairs, taken from one of the test problems, 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
```

The data suggest that the limit on the number of nodes on each ring will have more influence on the solution than the limit on demand. Hence, initially the level of demand between pairs of nodes will be ignored. At worst, the solutions found will give a lower bound on the minimum number of ADMs possible when the demand levels are taken into account.

For the instance just given, an optimal solution ignoring the traffic levels uses 22 ADMs on 4 rings. The sets of nodes installed on the rings are: {4, 8, 10, 12, 13}, {4, 5, 6, 11}, {1, 2, 9, 11, 13}, {7, 9, 12}. However, this is not a feasible solution if the level of demand is taken into account, because the third ring requires 41 channels, and 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 for this instance respecting the traffic limit uses 23 ADMs, as will be seen below.

¹ These are not used in their paper, but were sent to me by Cole Smith, along with the larger instances

3 A CSP Model with Integer and Set Variables

Sherali and Smith use 0/1 variables in modelling the SONET problem:

$$x_{ik} = 1 \text{ if node } i \text{ is assigned to ring } k, 1 \leq i \leq n, 1 \leq k \leq m \\ 0 \text{ otherwise}$$

where n is the number of nodes and m is the number of available rings.

It is of course possible to model the problem as a CSP using only these variables. However, the resulting CSP can only be solved in any reasonable time for the 15 smallest instances (with 7 nodes, 8 demand pairs, 4 available rings, and a maximum of 4 nodes allowed on each ring). It is in any case difficult to model the constraint that if there is a demand between a pair of nodes, they must both appear on the same ring, and 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 , and N_i is the set of rings that node i is assigned to. The x_{ik} variables are still used as the search variables.

The constraints are:

- $|R_k| \leq r$ where r is the maximum number of nodes on each ring, i.e. 5. The constraint could alternatively be expressed as: $\sum_i x_{ik} \leq r$.
- if there is a demand between nodes i and j , then $|N_i \cap N_j| \geq 1$ (i.e. there must be at least one ring that they are both assigned to).
- channelling constraints: $x_{ik} = 1$ iff $i \in R_k$ and iff $k \in N_i$.

The objective, the total number of ADMs in the network, is represented by an integer variable t , where $t = \sum_k |R_k|$. There is an implied constraint that $t = \sum_i |N_i|$. The optimal solution can be found using the optimization facilities in ILOG Solver: whenever a solution is found, Solver adds a constraint that in future solutions, the value of t must be smaller. Hence, when there is no solution satisfying the current constraint on t , the last solution found is known to be optimal.

Trying to construct good solutions by hand led to variable and value ordering heuristics which often find optimal or near-optimal solutions quickly. The next variable chosen is x_{ik} , where ring k is the smallest numbered ring not yet fully occupied and i (the node to place on this ring) is chosen according to the following criteria:

- if there is a node whose future (i.e. not yet accommodated) connections are all to nodes already on ring k , and those nodes in turn have no future connections other than to this node, choose it. (If the node is not allocated to ring k , it will have to be placed on another ring, along with its neighbouring nodes that are already on ring k , which is likely to be much more expensive.)
- if there is no such node, choose the node that is connected to the largest number of nodes already on ring k .

- break ties by choosing the node with largest future degree, defined to be the number of demand pairs involving the node that have not yet been accommodated on a ring. The tie-breaker is always used when starting a new ring, for instance.

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.

A number of implied constraints can be added to the model, as well as the one already mentioned, that $\sum_k |R_k| = \sum_i |N_i|$.

- if the degree δ_i of node $i \geq r$, it must be placed on more than one ring, i.e. $|N_i| > 1$;
- if two nodes i and j are connected, and each of them is connected to fewer than r other nodes (so that the first implied constraint does not apply), 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. $|N_i| + |N_j| \geq 3$.

As an example of the second type of implied constraint, suppose two nodes with a demand between them each have 2 other neighbours, and they have no neighbours in common, e.g. nodes 1 and 7 in Figure 1 below. If we consider each node in isolation, it seems that we could put it on just one ring. However, there must be a ring that both nodes are installed on, and there is not room for both sets of neighbouring nodes on the same ring. Hence, at least one of the nodes must also be on another ring.

The following constraints 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 $1 \leq k \leq m$;
- 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 $1 \leq k < l \leq m$. Otherwise there is an equally good solution in which the two rings are combined into one.

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

4 Symmetry Breaking

The available rings are indistinguishable: in any solution, the rings, with their associated nodes, can be permuted without changing the solution. 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. This symmetry can be eliminated using SBDS (Symmetry Breaking During Search) [3] by supplying symmetry functions describing the transpositions of pairs of rings/columns.

It would be possible to remove the symmetry between the rings in some other way, for instance by imposing ordering constraints between the sets of nodes on each ring. But it is important to ensure that symmetry-breaking constraints do not conflict with the variable ordering; it would be difficult to do this here, since the ordering is dynamic.

With SBDS and with the implied constraints and heuristics described earlier, the 15 medium-sized instances (10 nodes, 15 demand pairs, 6 rings and a maximum of 5 nodes allowed on each ring) are solvable in a reasonable time. They can be solved, even without breaking the symmetry, if the number of available rings is restricted to 4, instead of 6. Four rings are in fact sufficient, since none of the optimal solutions require more. These results are shown in Table 1: for the instances which are easiest to solve, eliminating the symmetry makes little difference, if any, especially in finding the optimal solution. For the most difficult instances, it reduces search overall by an order of magnitude.

Table 1. Solving medium-sized SONENT instances, allowing only 4 rings, with and without symmetry-breaking, using ILOG Solver. ‘Value’ is the minimum number of ADMs required. F is the number of backtracks (fails) to find the optimal solution, P is the total number of backtracks to prove optimality. Time is the cpu time in seconds on a 600MHz Celeron PC.

Instance	Value	With SBDS			No symmetry breaking		
		F	P	Time	F	P	Time
1	14	1	15	0.11	1	15	0.07
2	14	1	25,044	11.6	1	188,664	52.4
3	14	24	103	0.14	24	324	0.21
4	13	1,618	27,264	12.8	3,922	172,315	58.2
5	15	1	82,531	40.6	1	745,525	287
6	14	1	36,901	16.8	1	279,085	91.6
7	13	922	933	0.81	2,126	2,137	1.33
8	14	1,657	28,206	12.4	4,180	187,902	52.9
9	15	36,395	71,960	41.3	133,684	507,068	196
10	14	374	435	0.36	983	1,256	1.03
11	12	320	331	0.35	751	762	0.91
12	15	1,021	117,855	62.3	4,976	1,292,875	553
13	15	1,870	89,117	47.0	10,770	1,018,949	433
14	15	9	71,127	35.5	9	908,497	387
15	15	4	53,851	26.0	5	468,969	180

5 Faster Proofs of Optimality

Figure 1 shows the demand graph of one of the medium sized instances (instance 15). An optimal solution for this instance has 15 ADMs on 3 rings, represented by the sets $\{2,3,6,8,9\}$, $\{1,3,4,7,9\}$, $\{2,5,6,7,10\}$, i.e. the value of the objective variable t is 15. This solution is found in 4 backtracks, using the model described

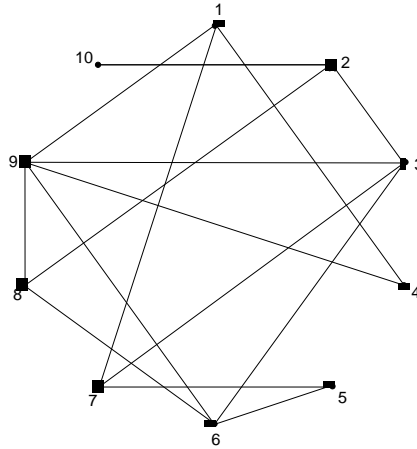


Fig. 1. Demand pairs in a sample instance of the SONENT problem

in the last section, but proving that it is optimal takes 53,581 backtracks. (This demonstrates that in optimization problems, it is worthwhile to record separately the effort required to *find* the optimal solution for the first time as well as the overall search effort, including *proving* that this solution is optimal. When trying to improve performance, it is useful to know whether most effort is going into finding good enough solutions, or into the proof of optimality, as here.)

How could we prove optimality more quickly? To prove that there is no solution with value 14, i.e. with $t = 14$, we attempt to find such a solution. To do that we need to try to reduce the number of nodes that appear on more than one ring. Node 9 must appear at least twice (because its degree in the demand graph is 5, and there are at most 5 nodes on each ring). There are also implied constraints on several pairs of nodes, specifying that at least one of the pair must appear at least twice, e.g. 1 and 7, 2 and 3. In the solution we are considering, most of these pairs already appear the minimum number of times; for instance, node 1 appears once and node 7 twice. The only exception is nodes 2 and 3, which both appear twice. To reduce the number of ADMs, we must find a solution where either node 2 or node 3 only appear once.

Suppose we introduce new integer variables n_i , where $n_i = |N_i|$. (Recall that N_i is a set variable representing the rings that node i is assigned to, and that $t = \sum_i |N_i|$.) n_i is the number of ADMs used for node i , i.e. the number of times that node i appears in the solution. These variables can be used to prove that the solution given earlier is optimal. First, the constraint that $t < 15$ is imposed, so that only solutions better than the one already found will be accepted. Adding the constraint $n_2 < 2$ leads to an immediate fail, so $n_2 \geq 2$. If the constraints $n_2 \geq 2$ and $n_3 < 2$ are imposed, there is no solution: the proof of this takes 122 fails. Hence we cannot have either node 2 or node 3 appearing just once in a solution with value < 15 , and so cannot reduce the number of nodes in the

solution: hence, the solution given earlier has been proved optimal, with little search.

The proof just outlined resulted from adding constraints to the model by hand. To automate the process, the n_i variables are used as search variables, and values are assigned first to them and then to the x_{ik} variables, i.e. the search first decides how many rings each node is on, and then decides which nodes are on each ring. This is still a complete algorithm; if no consistent assignment is found to the x_{ik} variables, the search will reassign the n_i variables.

To avoid assignments to the n_i variables which cannot be extended to a consistent solution, additional upper bounds on the n_i variables are added to the model; if the degree of node i is δ_i , it must appear on no more than δ_i rings, i.e. $n_i \leq \delta_i$. Note that this would not be a legitimate constraint if the demand levels were being taken into account.

Furthermore, since values are being directly assigned to the n_i variables, the following constraints can be added:

- 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 A_i is the set of neighbours of node i :

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

- 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.

$$\text{if } n_i = 2 \text{ and } x_{ik} = 1 \text{ and } x_{il} = 1 \text{ then } \{i\} \cup A_i \subseteq R_k \cup R_l \quad (2) \\ \text{for } 1 \leq i \leq n, 1 \leq k < l \leq m$$

These constraints help to complete the solution quickly or decide that it cannot be completed, once an assignment to the n_i variables has been made.

The symmetry between the rings can be broken in the same way as before, 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. With the two-phase model, 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.

The variable ordering heuristics used for the two sets of search variables are discussed separately below in section 8.

With the two-phase search just described, the medium-sized instances can be solved easily, as shown in Table 2. These are the same instances as in Table 1, except that the number of rings available is 6, as in the original data, instead of 4, which makes them harder to solve. Even so, with SBDS, these instances are solved much more quickly than with the simpler model used in Table 1. As before, symmetry breaking is clearly essential to being able to solve these problems.

Each of the largest instances (i.e. those used in Sherali & Smith's paper) can now easily be solved in under 25 seconds, provided that the number of available

Table 2. Solving medium-sized SONET instances with two-phase search, with and without symmetry-breaking.

Instance	Value	With SBDS			No symmetry breaking		
		F	P	Time	F	P	Time
1	14	13	21	0.62	553	561	2.27
2	14	1	13	0.10	1	13	0.06
3	14	16	29	0.64	241	251	1.34
4	13	1,320	1,338	6.98	>50K	-	-
5	15	156	174	1.24	>50K	-	-
6	14	0	14	0.57	0	14	0.10
7	13	4	16	0.55	13	25	0.11
8	14	1	13	0.56	1	13	0.09
9	15	7	26	0.63	19	715	2.43
10	14	707	715	3.74	40,989	40,997	118
11	12	1	12	0.54	1	12	0.08
12	15	0	25	0.61	0	590	2.27
13	15	32	63	0.90	5,366	5,965	19.2
14	15	398	498	2.85	30,169	38,083	128
15	15	2	17	0.57	2	588	2.36

rings is restricted to 5 and the demand level between pairs of nodes is still ignored. It is now time to reconsider the original problem, and in particular the demand capacity of the rings.

6 Modelling demand capacity

Sherali & Smith modelled the demand capacity of the rings by introducing a variable f_{kij} for every pair of nodes i, j with a traffic demand between them, and every ring k : f_{kij} is the fraction of the demand between nodes i and j that is assigned to ring k ($0 \leq f_{kij} \leq 1$). This gives a mixed integer programming model. These variables are used to model the fact that if a pair of nodes occurs on more than one ring, the demand between the nodes can be split over these rings.

Using fractional variables seems an unpromising approach for constraint programming, and moreover introduces alternative solutions. For instance, the optimal solution for the example in section 2, taking the demand levels into account, has a ring with the set of nodes $\{2,3,7,10,13\}$. The total traffic demand between these nodes 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 another ring with enough spare capacity to take all the demand between these two nodes. Hence, a feasible solution can allocate either 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 second ring.

To avoid exploring equivalent alternative solutions during search, we should avoid making the decision about how the demand is split between the rings, and

simply ensure that it can be done feasibly by adding further constraints to the existing model.

First, more variables are needed, relating to the edges in the demand graph, to allow the constraints to be expressed. Let

$$w_{pk} = 1 \text{ if edge } p \text{ is on ring } k, 1 \leq l \leq e, 1 \leq k \leq m \\ 0 \text{ otherwise}$$

If edge p links nodes i and j , then we have the following channelling constraints: $w_{pk} = 1$ iff $x_{ik} = 1$ & $x_{jk} = 1$, for every edge p . Also let E_k be the set of edges on ring k , giving the further channelling constraints: $w_{pk} = 1$ iff $p \in E_k$.

We could insist that the total demand for channels from the edges on any ring must be no more than the capacity of the ring, i.e.

$$\sum_{p:p \in E_k} d_p \leq b \quad 1 \leq k \leq m \quad (3)$$

where d_p is the traffic demand on edge p , and b is the traffic capacity of a ring.

However, this constraint is too tight and would not allow the demand between a pair of nodes to be split between two or more rings. Hence, although any solution found that satisfies this constraint is feasible, it may not be optimal.

It is clearly necessary that if a demand pair occurs on only one ring, the demand between the nodes must fit onto that ring, and a ring must accommodate all the demands that cannot be assigned to any other ring. Hence, for any ring, the sum of the demands between demand pairs that occur only on that ring must be no more than the capacity of the ring:

$$\sum_{p,p \in E_k; p \notin E_l, l \neq k} d_p \leq b \quad 1 \leq k \leq m \quad (4)$$

Although these constraints allow a demand pair to be on more than one ring, they do nothing to check that the total demand between the pair can be accommodated along with the other demand pairs on these rings. Hence the constraints are too loose and may allow infeasible solutions.

We keep the constraints (4) and add constraints on pairs of rings. If a demand pair occurs on two rings, the sum of the demands that occur only on those two rings must be no more than twice the ring capacity:

$$\sum_{p,p \in E_k \cup E_l; p \notin E_h, h \neq k,l} d_p \leq 2b \quad 1 \leq k < l \leq m, E_k \cap E_l \neq \emptyset \quad (5)$$

The model is extended to include constraints (4) and (5) for every ring and for every pair of rings. Although tighter than constraints (4) alone, they might still conceivably allow infeasible solutions, since there may be demand pairs on either or both of the rings k, l which are also on other rings and so not included in the constraints. To prevent this the constraints could be extended to sets of three rings, and so on. However, in practice, for the instances considered, the

optimal solutions found are feasible, and the demands will fit onto the rings as allocated.

Some of the implied constraints added earlier must also be modified when the demand capacity of the rings is taken into account. For instance, 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 δ_i should appear on at most δ_i rings, and this constraint is dropped.

Having modified the model to handle the level of demand between nodes, the large instances can be solved. First, an upper bound on the new optimal value is found. This can be done easily; the existing solutions only violate the demand capacity on one or two rings, and the infeasibility can be fixed by duplicating one or two demand pairs on a new ring, or an existing ring with spare capacity. (Currently, this is done by hand but the process could be automated.) This gives a feasible solution using only a few more nodes than the solution already found, and hence an upper bound on the value of the new optimum.

Table 3 shows, in the left-hand columns, the performance of the two-phase model in solving the original Sherali & Smith instances (13 nodes, 24 demand pairs, ring capacity 5 nodes and 40 traffic channels, and 7 rings available) if the traffic capacity of the rings is ignored. For six of these 15 instances, the solutions found are still feasible (and therefore optimal) if the traffic between each pair of nodes and the traffic capacity of each ring is taken into account. In the other cases, the optimal solution already found gives lower and upper bounds on the new optimum, and the right-hand columns show the performance of the two-phase search in re-optimizing, using these bounds to limit the search. The average solution time is 167 sec. (including both finding the optimal solution ignoring the demand levels and then if necessary re-optimizing using upper and lower bounds derived from this solution).

7 The Objective as a Search Variable

The final improvement in performance arose from examining the assignments to the n_i variables considered during search. When a variable n_i is chosen as the next variable to assign, each value in its domain is tried in turn. Before a solution has been found, values up to $\min(\delta_i, m)$ will be tried. However, if the degree δ_i of node i is large, for the largest values in a variable's domain, $\sum_i n_i$ will be larger than in the optimal solution. Considering such assignments is ultimately wasted effort. This leads to the idea of considering assignments for which the objective variable t has a specified value, t_0 , and only increasing t_0 once every assignment to the n_i variables for which $\sum_i n_i = t_0$ has been considered without success. This is easily implemented by making t a search variable and choosing it first.

In this way, the search will never consider assignments in which $\sum_i n_i$ is more than the optimal value; indeed, the first solution found is guaranteed to be optimal. Every assignment in which $\sum_i n_i$ is less than the optimum must still

Table 3. Solving large SONET instances, with a two-phase search process. Dashes in the right-hand columns indicate that the solution already found ignoring the demand capacity of the rings is still optimal when it is taken into account.

Instance	Without demands				With demands				
	Optimal	F	P	Time	Initial bound	Optimal	F	P	Time
1	22	32	1,424	12.8	24	22	3,864	3,883	69.2
2	20	11,988	12,004	122.4	-	20	-	-	-
3	22	427	1,370	14.8	-	22	-	-	-
4	23	19,824	24,186	196	25	23	8,668	8,686	144
5	20	221	234	2.85	22	22	17	1,992	39.0
6	22	148	1,200	10.5	-	22	-	-	-
7	20	61,492	61,514	451	22	20	1,685	6,534	127
8	20	2,222	2,256	26.4	-	20	-	-	-
9	22	6,144	7,480	57.5	26	23	18,218	25,932	421
10	23	3,252	5,944	52.4	25	24	1,056	2,708	57.9
11	22	5,220	5,322	44.7	-	22	-	-	-
12	20	914	944	8.98	24	22	249	1,209	43.3
13	21	1,506	2,998	26.2	-	21	-	-	-
14	23	405	1,154	11.8	25	23	3,130	3,145	59.6
15	22	41,277	41,349	416	23	23	3,312	4,284	83.3

be considered, as before, in order to prove optimality. In the previous model, all such assignments which have not previously been encountered during the search are considered once the optimal solution is found, in order to complete the proof of optimality. Now, all these assignments will already have been met by the time the optimal solution is found.

This would not be a useful solution strategy if finding the optimal solution were likely to be very difficult and we were prepared to settle for a good solution and forgo the proof of optimality. 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) and the optimal value is between 20 and 23. If the gap were much larger, proving optimality might be unrealistic.

Table 4 shows the results with this strategy. It is no longer necessary to derive an upper bound on the optimal solution with demand levels from the optimal solution ignoring them. The optimal solution with demand levels was found in all cases, even when the solution already found was still feasible. It is usually very quick to find the same solution again in these cases, and avoids checking the solution for feasibility by hand (although an automatic check would be quicker).

The average solution time for the instances used in the Sherali & Smith paper is now 63 sec. (including an average of 15 sec. to find the optimal solution ignoring the demand levels). In [6], the average solution time (with the level of symmetry breaking giving best performance) was 21,000 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 mixed integer LP model, but it is clear that these instances are now relatively easy to solve using the CP model and search strategy described.

On the other hand, the success of the CP model does to some extent depend on the characteristics of the set of instances. In particular, the solution ignoring the level of demand is not too far from the true optimal solution and constraints (4) and (5) in section 6 are sufficient to give a feasible solution.

Table 4. Solving large SONET instances, with the objective as the primary search variable. * indicates that the optimal solution without taking the demand levels into account is also feasible for the full problem.

Instance	Without demands			With demands		
	Optimal	F	Time	Optimal	F	Time
1	22	1,425	15.8	22	442	9.14
2	20	61	1.27	20*	52	1.90
3	22	1,065	13.3	22*	72	2.44
4	23	5,722	58.8	23	6,147	101
5	20	192	2.32	22	2,036	42.6
6	22	1,148	12.9	22*	25	0.69
7	20	151	2.12	20	6,525	128
8	20	118	2.14	20*	49	1.78
9	22	1,626	18.1	23	12,154	219
10	23	3,252	41.8	24	2,516	57.0
11	22	263	3.85	22*	90	2.80
12	20	183	2.67	22	1,135	22.7
13	21	2,637	29.0	21*	673	14.6
14	23	938	12.1	23	1,257	24.8
15	22	668	9.06	23	4,271	88.5

8 Variable Ordering for Two-Stage Search

For the two-stage search, with two separate sets of search variables being assigned consecutively, two variable ordering heuristics are needed. The results already given for these models (Tables 2, 3 and 4) depend on the variable ordering heuristics used. The same heuristics have been used for the initial two-stage search, described in section 5, and also for the variants that deal with the demand capacities of the rings and/or choose the value of the objective variables first.

A number of variable ordering heuristics for the first phase, i.e. for assigning the n_i variables, have been tried. However, the comparison is clearer in the final model, ignoring the demand capacity. In that model, three heuristics have been compared: smallest domain, minimum degree and maximum degree.

Any variable ordering will generate the same set of sub-optimal complete assignments to the n_i variables to pass to the second stage of search (i.e. assign-

ments for which the value of t is less than its optimal value), since all 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, but 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 much worse, and smallest domain slightly worse, than maximum degree. Evidently the total number of complete assignments to the n_i variables at the optimal value of t is again the same for all three heuristics, but they each consider them in a different order, and hence find a solution earlier or later. Its poor performance indicates that the assignments considered first by minimum degree are less likely to be successfully extended to the x_{ik} variables than those considered first by maximum degree, but it is difficult to identify the reason.

Maximum degree also compares well with the other two variable ordering heuristics for the other variants of the two-stage search, i.e. for the previous model with the standard form of optimization and for both models with the constraints to take into account the demand capacities of the rings.

A number of variable ordering heuristics have been investigated for the x_{ik} variables, which are assigned once the a complete assignment to the n_i variables have 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 best performance was found by choosing 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. Given the set of constraints (1) in section 5, all its neighbours will then be placed on the same ring; by choosing the node with maximum degree, the largest number of neighbouring nodes will be placed. Since these nodes must be on the same ring, postponing placing them is likely to lead to future failure and wasted search.

The heuristics investigated were selected largely by trial and error, and so it is very likely that better heuristics exist. The choice, especially in the first stage, 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. Hence, although the second stage variables do not directly influence the first stage of search, they have an indirect influence on the performance of variable ordering heuristics that is hard to take into account.

9 Discussion

The first simple CP model for the SONET problem could only solve very small instances; after a lot of effort redesigning the model and devising better search strategies, much larger instances can now be solved comfortably.

The two-phase search, in the second model presented, is novel and could be useful in other contexts. Here, the search first decides how many rings each node should be installed on and then decides which rings they should be. A similar idea was used for a difficult instance of the template design problem [4]. The common features of the two-phase search in the two problems are:

- the first phase assigns values to a set of ‘intermediate’ variables. These are closer to the objective than the original variables, in the sense that an assignment to an intermediate variable can propagate to the objective. Conversely, tightening the bounds on the objective can prune the domains of the intermediate variables. In contrast, the original variables and the objective have very little direct effect on each other;
- in both the template design problem and the SONET problem, the objective is the sum of the intermediate variables. However, other relationships might also be possible;
- an assignment to the intermediate variables does not give a complete solution to the problem, but it does make finding a complete solution (or proving that there is no solution) much easier.

The intermediate variables can be thought of as a bridge between the objective variable and the original decision variables, allowing a new bound on the objective to propagate through to the decision variables. This could be a useful strategy in other cases where the objective can, in effect, be decomposed into a set of intermediate variables.

An extension to the two-stage search minimizes the objective by considering progressively increasing values, proving that each value cannot be attained before going on to the next. This means that the first solution found is guaranteed to be optimal. This is in contrast to the usual branch-and-bound approach where an incumbent solution is improved until no further improvement can be made, and proving optimality often requires extensive search after the optimal solution is found. The approach adopted here could be useful for other cases where the objective has only a small range of possible values.

Another feature of modelling the SONET problem that is found in modelling other problems is that the variable ordering heuristics have a profound effect on the performance. As noted earlier, the variable ordering heuristics used were derived by trial and error. Currently, we do not know enough about how to choose variable ordering heuristics; much of what we do know is based on studies on random binary constraint satisfaction problems, and may not transfer to CSPs derived from real problems if the constraints are neither binary nor uniform. Standard heuristics may be still less applicable to optimization problems and the two-stage search adopted for this problem complicates matters further. Previous experience has shown that heuristics which are a good choice when trying to find a good solution may be much less effective when it comes to proving optimality [7]; in general, heuristics for optimization problems have to do both. In order to be able to choose good heuristics for optimization problems, we will either need a better understanding of these issues, or a systematic way of choosing

heuristics tailored to specific problems (as in [1], for instance); preferably, both. It is quite possible that much better heuristics for the SONET problem could then be found and that larger problems could be solved more quickly.

Considerable effort has been put into remodelling the problem; several iterations were needed before the larger problems could be solved satisfactorily. 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.* [2], using models of the SONET problem as an illustration.

Remodelling an initially intractable problem has eventually resulted in a CP model that appears competitive with Sherali & Smith's mixed integer LP model. Optimization is generally considered to be difficult for CP, but more effort put into CP modelling could pay off in other optimization problems too. Régin [5], 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. The author is a member of the APES group (see <http://www.dcs.st-and.ac.uk/~apes>) and would like to thank the other members for their continuing 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 work was supported by EPSRC grant GR/M90641.

References

1. S. L. Epstein, E. C. Freuder, R. Wallace, A. Morozov, and B. Samuels. The Adaptive Constraint Engine. In P. van Hentenryck, editor, *Principles and Practice of Constraint Programming - CP 2002*, LNCS 2470, pages 525–540. Springer, 2002.
2. 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. Available from <http://www-users.cs.york.ac.uk/~frisch/Reformulation/02/Proceedings/>.
3. 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.
4. L. G. Proll and B. M. Smith. ILP and Constraint Programming Approaches to a Template Design Problem. *INFORMS Journal on Computing*, 10:265–275, 1998.
5. 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. Springer, 2003.
6. H. D. Sherali and J. C. Smith. Improving Discrete Model Representations Via Symmetry Considerations. *Management Science*, 47:1396–1407, 2001.
7. B. M. Smith, K. E. Petrie, and I. P. Gent. Models and Symmetry breaking for 'Peaceable Armies of Queens'. Technical Report APES-50-2002, APES Research Group, May 2002. Available from <http://www.dcs.st-and.ac.uk/~apes/apesreports.html>.