Pergamon

S0305-0548(96)00001-9

Computers Ops Res. Vol. 23, No. 9, pp. 845–856, 1996
Copyright © 1996 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0305-0548/96 $15.00 + 0.00

ORGANIZING A SOCIAL EVENT — A DIFFICULT PROBLEM OF COMBINATORIAL OPTIMIZATION

S.C. Brailsford,[1]†‡ P.M. Hubbard,[1]§ B.M. Smith[2]¶ and H.P. Williams[1]||

[1] Faculty of Mathematical Studies, University of Southampton, SO17 1BJ, and [2] Department of Computer Science, University of Leeds, LS2 9JT, U.K.

(Received March 1995; in revised form November 1995)

Scope and Purpose—The purpose of this paper is to contrast two techniques, linear integer programming and constraint programming, for the solution of discrete optimization problems. Integer programming is well known, but we argue that constraint programming is a powerful technique which is still relatively unknown to the mainstream operations research community yet has much to offer in the solution of combinatorial optimization problems. The interesting and unusual problem we consider arose in the course of organizing the social programme at a yacht rally. The problem proved intractable to integer programming alone, but a combination of linear programming and constraint programming found a solution remarkably quickly. In this paper we describe the application of constraint programming to this problem, and we identify and discuss the features of this problem which account for the success of this combined approach.

Abstract—This interesting problem arose in the context of organizing a "progressive party" at a yachting rally. Some yachts were to be designated hosts; the crews of the remaining yachts would then visit the hosts for six successive half-hour periods. A guest crew could not revisit the same host, and two guest crews could not meet more than once during the evening. Additional constraints were imposed by the capacities of the host yachts and the crew sizes of the guests. Although this problem has many of the classical features of combinatorial optimization problems, it does not belong to any previously known class of problem. Integer programming formulations which included all the constraints resulted in very large models and all attempts to find a solution failed. However by solving a simple relaxation of the problem using linear programming we obtained a lower bound for the solution, which combined with a feasible solution obtained (spectacularly easily) by constraint programming led to an optimal solution. We describe our computational experience and discuss the features of this problem which account for the failure and success of the two approaches. Copyright © 1996 Elsevier Science Ltd

1. INTRODUCTION

This seemingly frivolous-sounding problem arose in the context of organizing the social program for a yachting rally. One of the early events at the rally was designed to enable people to get to meet as many of the other attendees as possible. The 39 yachts at the rally were all moored in a marina on the Isle of Wight; they were of different sizes and had varying crews, ranging from 1 to 7. An evening party was planned at which some of these boats were to be designated host boats. The crews of the remaining boats would visit the host boats in turn for six successive half-hour periods during the evening. The crew of a guest boat would stay together as a unit for the whole evening, and the crew of a host boat would remain on board to act as hosts. A guest crew could not revisit the same host boat, and guest crews could not meet more than once during the course of the evening. Additional capacity constraints were imposed by the sizes of the host boats.

† Sally C. Brailsford is a Lecturer in Operational Research at the University of Southampton, U.K. where she received her Ph.D. in Operational Research. Her research interests include applications of logic to integer programming, simulation, and modelling of disease, in particular AIDS.

‡ To whom correspondence should be addressed.

§ Peter M. Hubbard is a Computer Officer at the University of Southampton. He is a keen amateur yachtsman.

¶ Barbara M. Smith is a Senior Lecturer at the University of Leeds, where she received her Ph.D. in Operational Research. Her current research is in Artificial Intelligence: her research interests include constraint satisfaction problems and the applications of constraint programming, particularly in scheduling.

|| H. Paul Williams is Professor of Operational Research at the University of Southampton, U.K. He is well known for his work on Mathematical Programming and as the author of two books, *Model Building in Mathamatical Programming* and *Model Solving in Mathematical Programming*, in addition to many journal articles.

The problem facing the rally organizer was that of minimizing the number of host boats, since each host had to be supplied with food and other party prerequisites. The number of potential feasible solutions was relatively small, since the problem was highly constrained.

There were a number of complicating factors in the real-life problem. For example, the rally organizer's boat was constrained to be a host boat, although it had a relatively small capacity, because he had to be readily available to deal with emergencies. Two other boats had crews consisting of parents with teenage children, and these boats were also constrained to be host boats; the crews split up so that the parents remained on board the host boat and the children became a "virtual boat" with capacity of zero. The rally organizer's children formed a third "virtual boat"; thus the total number of boats was 42. The data for this problem is given in Table 1.

2. THEORETICAL APPROACH TO THE UNCAPACITATED PROBLEM

If we ignore the capacity constraints, we can prove the following result for the general case of n boats and t time periods.

Theorem:
If $t = 1$, the number of hosts required is 1, for any n. If $t > 1$, then the number of hosts required is greater than or equal to

$$s = \max\{t, \lceil\sqrt{n}\rceil\}.$$

Moreover, if s is prime, then this bound can be attained.

Proof: The case $t = 1$ is obvious. Let $t > 1$, and let the total number of hosts be H. For period k, denote by m_k the maximum number of guests on any one of these hosts. Then in period $k + 1$, each

Table 1. The data

Boat	Capacity	Crew
1	6	2
2	8	2
3	12	2
4	12	2
5	12	4
6	12	4
7	12	4
8	10	1
9	10	2
10	10	2
11	10	2
12	10	3
13	8	4
14	8	2
15	8	3
16	12	6
17	8	2
18	8	2
19	8	4
20	8	2
21	8	4
22	8	5
23	7	4
24	7	4
25	7	2
26	7	2
27	7	4
28	7	5
29	6	2
30	6	4
31	6	2
32	6	2
33	6	2
34	6	2
35	6	2
36	6	2
37	6	4
38	6	5
39	9	7
40	0	2
41	0	3
42	0	4

of these m_k guests must go to a different host, and so we need at least $m_k + 1$ hosts in total. In theory, any one of the $m_k + 1$ hosts could accommodate m_k guests in period k, and each of these guests could go to a different host in period $k + 1$ without meeting again, so for more than two periods we have an upper bound of $m_k(m_k + 1)$ for the total number of guests which can be accommodated by $m_k + 1$ hosts.

In our notation, the total number of guests is $n - H$ and $m_k + 1 = H$, so we have $n - H \leqslant H(H - 1)$ and thus $H \geqslant \lceil \sqrt{n} \rceil$. Clearly, $H \geqslant t$ for all t.

In our case $s = \max\{6, \lceil \sqrt{42} \rceil\} = 7$. In Table 2 we give a feasible solution to the uncapacitated problem with precisely 7 host boats. For the sake of clarity, we have labelled the guest boats 1 ... 35 and the hosts as $H1$, $H2$ etc.

To prove this is feasible, we write $g = i + 7j$, where $i = 1 \ldots 7$ and $j = 0 \ldots 4$; thus $g = 1 \ldots 35$ are the guests and we have five sets of seven guest boats, where j denotes the group a guest is in and i denotes its index within that group. At time Tm, each guest g visits host Hn, where

$$n = i - (j + 1)(m - 1) \bmod (7)$$

It can be seen from Table 1 that the guests visit the hosts in a cyclic order, moving up the list of hosts in each successive time period.

To prove that a guest does not revisit the same host, suppose that g visits Hn at times Tm and Tm'. Then

$$i - (j + 1)(m - 1) = i - (j + 1)(m' - 1) \bmod (7)$$

and thus

$$(j + 1)(m - 1) = (j + 1)(m' - 1) \bmod (7)$$

Since $(j + 1)$ and 7 are coprime (i.e. have no common factors) and $j + 1 \neq 0$, we can divide through by $(j + 1)$; thus we obtain $(m - 1) = (m' - 1) \bmod (7)$ and thus $m = m'$.

To check that two guests cannot meet more than once, suppose that two guests g and $g' = i' + 7j'$ meet at times Tm and Tm'. Then

$$i - (j + 1)(m - 1) = i' - (j' + 1)(m - 1) \bmod (7)$$

and

$$i - (j + 1)(m' - 1) = i' - (j' + 1)(m' - 1) \bmod (7).$$

Subtracting, we obtain

$$(j + 1)(m - m') = (j' + 1)(m - m') \bmod (7)$$

and since $(m - m')$ and 7 are coprime and $m \neq m'$, we have $j = j' \bmod (7)$ and thus $j = j'$. Hence $i = i' \bmod (7)$, giving $i = i'$, and thus $g = g'$.

It can easily be seen that this proof generalises to the case of any prime s.

However for the real-life problem, the capacity constraints are binding and the number of host boats required is at least 13. This was proved by solving the model TEST described in Section 4 for a single time period.

Table 2. A feasible solution to the uncapacitated problem with 42 boats

	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$
$H1$	1,8,15, 22,29	2,10,18, 26,34	3,12,21, 23,32	4,14,17, 27,30	5,9,20, 24,35	6,11,16 28,33
$H2$	2,9,16, 23,30	3,11,19, 27,35	4,13,15, 24,33	5,8,18, 28,31	6,10,21, 25,29	7,12,17, 22,34
$H3$	3,10,17, 24,31	4,12,20, 28,29	5,14,16, 25,34	6,9,19, 22,32	7,11,15, 26,30	1,13,18, 23,35
$H4$	4,11,18, 25,32	5,13,21, 22,30	6,8,17, 26,35	7,10,20, 23,33	1,12,16 27,31	2,14,19, 24,29
$H5$	5,12,19, 26,33	6,14,15, 23,31	7,9,18, 27,29	1,11,21, 24,34	2,13,17, 28,32	3,8,20, 25,30
$H6$	6,13,20, 27,34	7,8,16, 24,32	1,10,19, 28,30	2,12,15, 25,35	3,14,18, 22,33	4,9,21, 26,31
$H7$	7,14,21, 28,35	1,9,17, 25,33	2,11,20, 22,31	3,13,16, 26,29	4,8,19, 23,34	5,10,15, 27,32

3. OBTAINING AN UPPER BOUND HEURISTICALLY

The rally organiser devised a heuristic method of solution which involved firstly ordering the boats in decreasing order of available space, i.e. total capacity minus crew size. The first k boats were then fixed to be host boats and the remaining $42 - k$ were successively allocated as guests in order. If it proved to be impossible to allocate a boat for a particular time period, that boat was given a *wildcard*, which in practice meant that they were given a bottle of wine and had to return their own boat for half an hour. This "greedy" heuristic was programmed in Turbo Pascal. The resulting program ran in less than a minute for each value of k. The least number of host boats which did not involve giving any wildcards was found to be 17.

4. OBTAINING A LOWER BOUND BY LINEAR PROGRAMMING

The idea of checking feasibility by counting up the total number of available spaces on the host boats, and comparing this with the total number of guests, led to the formulation of the following model TEST, which is clearly a considerable relaxation of the original problem. Here we only consider a single time period, and moreover we do not require that crews stay together; we simply demand that the total capacity of all the hosts cannot be exceeded. Given data arrays *crew*(i) and *cap*(i) of the crew sizes and capacities respectively, we define variables $H(i)$ and $G(i)$ as follows:

$$\begin{aligned} H(i) &= 1 \text{ iff boat } i \text{ is chosen to be a host} \\ &= 0 \text{ otherwise} \end{aligned}$$

$$\begin{aligned} G(i) &= 1 \text{ iff boat } i \text{ is chosen to be a guest} \\ &= 0 \text{ otherwise} \end{aligned}$$

and we choose as an objective to minimize the total number of hosts. We have 42 constraints saying that a boat must be either a guest or a host:

$$H(i) + G(i) = 1, i = 1, \ldots, 42$$

and one constraint saying that the total capacity cannot be exceeded:

$$\sum_{i=1}^{42} \{crew(i).H(i) + crew(i).G(i) - cap(i).H(i)\} \leqslant 0$$

This model has 84 variables and 43 constraints. The LP relaxation solved in less than 1 second (28 simplex iterations) using the XPRESSMP optimizer [1] on an IBM 486 DX PC. The objective value was 12.2, giving a lower bound of 13 for the solution of the original problem.

It is interesting to note that for this particular data, the same result can be obtained by the following argument; if we order the boats in decreasing order of total capacity, irrespective of their crew size, then the first 12 boats cannot accommodate all 42 crews, whereas the first 13 can.

5. FIRST INTEGER PROGRAMMING FORMULATION OF THE COMPLETE PROBLEM

The problem was formulated as a zero-one integer programme. Variables were chosen as follows:

$$\begin{aligned} \delta_i &= 1 \text{ iff boat } i \text{ is used as a host boat} \\ &= 0 \text{ otherwise} \\ \gamma_{ikt} &= 1 \text{ iff boat } k \text{ is a guest of boat } i \text{ in period } t \\ &= 0 \text{ otherwise} \end{aligned}$$

The constants c_i and K_i denote respectively the crew size and the capacity of boat i. We did not attempt to model the "splitting" of any boats but assumed that all boats were either guests or hosts. The rally organizer's boat was constrained to be a host in all models.

With these variables, a preliminary "naive" formulation was devised:

5.1. Objective

$$\text{Minimize } \sum_i \delta_i, \text{ subject to :}$$

Constraints CD. A boat can only be visited if it is a host boat:

$$\gamma_{ikt} - \delta_i \leqslant 0 \qquad \text{for all } i, k, t, i \neq k \tag{1}$$

Constraints CCAP. The capacity of a host boat cannot be exceeded:

$$\sum_k c_k \gamma_{ikt} \leqslant K_i - c_i \qquad \text{for all } i, t, i \neq k \tag{2}$$

Constraints GA. Each guest boat must always have a host:

$$\sum_i \gamma_{ikt} + \delta_k = 1 \qquad \text{for all } k, t, i \neq k \tag{3}$$

Constraints GB. A guest boat cannot visit a host boat more than once:

$$\sum_t \gamma_{ikt} \leqslant 1 \qquad \text{for all } i, k, i \neq k \tag{4}$$

Constraints W. Any pair of guests can meet at most once:

$$\gamma_{ikt} + \gamma_{ilt} + \gamma_{jks} + \gamma_{jls} \leqslant 3 \tag{5}$$

for all $i, j, k, l, t, s, i \neq j, i \neq k, k < l, i \neq l, k \neq j, j \neq l, s \neq t$

The formulation of the W constraints is explained as follows. If guests k and l meet on boat i in period t ($\gamma_{ikt} = \gamma_{ilt} = 1$), then they cannot meet again on boat j in period s ($\gamma_{jks} = \gamma_{jls} = 1$). Thus at most 3 of the variables $\gamma_{ikt}, \gamma_{ilt}, \gamma_{jks}, \gamma_{jls}$ can be non-zero.

The constraints W, which have six indices, clearly lead to an astronomical number of rows when the problem is modelled. The size of the model is $O(n^4 t^2)$, where n is the number of boats and t is the number of time periods. However this model has a moderate number of variables, namely $O(n^2 t)$.

In an attempt to reduce the size of the problem, we ordered the data in decreasing order of total capacity minus crew size. Because of the nature of the data there were clearly a large number of alternate solutions; we were not interested in differentiating between any of these. From the data, it was also clear that in any optimal solution there were some boats which would always be chosen to be hosts because of their large capacity and small crew size. By the same token, some boats would clearly never be chosen to be hosts: for example, the three "virtual boats" with zero capacity. We introduced parameters *hostmin* and *hostmax* such that the range of indices for potential hosts was restricted to 1 ... *hostmax* and the range of indices for potential guests was restricted to *hostmin* + 1 ... 42. We then solved the problem for different values of *hostmin* and *hostmax*, using our knowledge of the data and the heuristic solution, in order to put bounds on the objective value. This process is described in more detail later.

To test the formulation, we experimented with a reduced problem with 15 boats and 4 time periods, and we restricted the indices by setting *hostmin* = 4 (since we knew we needed at least 4 hosts) and *hostmax* = 8 (which seemed a probable overestimate of the required number of hosts). This resulted in a model with 379 variables and 18,212 rows. The LP relaxation solved in 259 s using the XPRESSMP optimizer on an IBM 486 DC PX, in 816 simplex iterations, giving an objective value of 3.42.

6. SECOND INTEGER PROGRAMMING FORMULATION

We introduced a further set of zero-one variables

$$x_{iklt} = 1 \text{ iff crews } k \text{ and } l \text{ meet on boat } i \text{ in time period } t$$
$$= 0 \text{ otherwise}$$

and replaced the constraints W by the three sets following sets S, V and Y. S and V together define the variables x in terms of the γ variables:

Constraints S:

$$2x_{iklt} - \gamma_{ikt} - \gamma_{ilt} \leqslant 0 \qquad \text{for all } i,k,l,t,k < l, i \neq k, i \neq l \tag{6}$$

In other words, $x_{iklt} = 1$ implies $\gamma_{ikt} = \gamma_{ilt} = 1$. Conversely, the condition $\gamma_{ikt} = \gamma_{ilt} = 1$ implies $x_{iklt} = 1$ is guaranteed by constraints V:

Constraints V:

$$\gamma_{ikt} + \gamma_{ilt} - x_{iklt} \leqslant 1 \qquad \text{for all } i,k,l,t,k < l, i \neq k, i \neq l \tag{7}$$

The Y constraints then replace the W constraints in the first formulation.

Constraints Y. No two guests can meet more than once:

$$\sum_t \sum_l x_{iklt} \leqslant 1 \qquad \text{for all } i,k,k < l \tag{8}$$

At the expense of introducing extra variables, we had now reduced the number of rows. The number of variables is now $O(n^3t)$ but the number of rows is reduced, also to $O(n^3t)$.

7. COMPUTATIONAL EXPERIMENTS ON REDUCED MODEL

The second formulation was used in a variety of computational experiments with the reduced 15-boat problem. As before, we set *hostmin* = 4 and *hostmax* = 8. The computational results are tabulated in Table 3. Firstly, the problem was solved directly as it stands (model PS1). Then a facility of the XPRESSMP package was used which enables certain constraints to be introduced only if a particular solution violates them. This greatly reduces the initial size of a model. This facility is called MVUB (Make Variable Upper Bounds) and applies only to constraints of the form

$$x - My \leqslant 0$$

The *CD* constraints were in this form already but the *S* constraints had to be disaggregated in order to use the MVUB facility. Normally disaggregation would result in a sharper LP relaxation (i.e. one with an objective value closer to that of the IP optimum), but since in this case all the coefficients of x_{ijkt} in the other constraints of the model (apart from the V constraints) are unity, it can be shown by Fourier–Motzkin elimination that this will not be the case [2]. In the second version of the model (PS11) the *CD* and *S* constraints were both modelled using MVUB; in the third (PS12) the *S* constraints were modelled explicitly and the *CD* constraints were modelled using MVUB.

Since the IP solution time was equal to the sum of the MVUB time and the time spent doing branch-and-bound, model PS1 solved fastest. Thus the MVUB feature was not helpful in this case.

Special ordered sets of type I were tried but also found not to be useful. A set of variables form a special ordered set of type I if at most one of them can be nonzero. This device is useful if there is some natural ordering on the variables, when it can reduce the time spent doing branch-and-bound. For example, the γ variables could be treated as special ordered sets; for each value of i and k, at most one of the set $\{\gamma_{ikt}, t = 1 \ldots 6\}$ can be nonzero. However for the x variables there was no

Table 3. Results for the 15-boat, 4-period model

Model	PS1	PS11	PS12
Rows	4386	2130	4022
Columns	2271	2271	2271
LP solution time (s)	101	16	19
Number of iterations	1474	696	497
LP objective value	3.42	3.42	3.42
MVUB time (s)	—	561	697
Branch-&-Bound time (s)	2113	1852	3789
Number of nodes	287	279	311
IP objective value	5.00	5.00	5.00

obvious ordering of the variables in the set, and in general this approach was not found to be helpful either.

It would also be possible to tighten the LP relaxation by adding extra "covering" constraints generated from the CCAP constraints. For example, if we have the capacity constraint

$$4\gamma_{121} + 2\gamma_{131} + 4\gamma_{141} + 3\gamma_{151} \leqslant 7$$

then we could add the covering constraints

$$\gamma_{121} + \gamma_{141} \leqslant 1$$

$$\gamma_{121} + \gamma_{131} + \gamma_{151} \leqslant 2$$

$$\gamma_{131} + \gamma_{141} + \gamma_{151} \leqslant 2$$

However there would be a vast number of such constraints and so this did not seem a particularly fruitful approach.

Another approach we tried was to omit the constraints 6, 7 and 8, solve the LP relaxation of the resulting problem and then add in cuts of the form

$$\sum_{k \in Q} \gamma_{ikt} - \sum_{k,l \in Q} x_{iklt} \leqslant \left\lfloor \frac{|Q|}{2} \right\rfloor$$

for index sets Q, where $|Q|$ is odd. We found by inspection that many of these were violated by the fractional solution. However automating the process of inspecting the solution, identifying the appropriate index sets and then generating the corresponding cuts would have been computationally prohibitive. Equally, there would be no advantage in generating all possible cuts (even for sets of cardinality 3 only), as this would simply have resulted in another enormous model.

We concluded that unfortunately our experiments with the reduced model did not indicate a successful solution strategy for the full model.

8. COMPUTATIONAL EXPERIMENTS ON LARGE MODEL

The size of the full model defeated all the modellers at our disposal, even using indices restricted by *hostmin* and *hostmax*. Therefore we adopted a heuristic approach to finding a solution. By the arguments described in Section 5, it appeared sensible to order the boats in decreasing order of "spare room", i.e. *cap*(i) – *crew*(i). However, it might not necessarily be sensible to have a large crew as a guest, irrespective of the spare capacity on such a boat, since the number of potential hosts for this crew would be limited (or even theoretically zero, if the crew were larger than any of the available host capacities). Nevertheless with this ordering, choosing the first 13 boats as hosts was numerically feasible (there were 4 spare places after all the guests had been accommodated) and the largest guest crew could be accommodated on all but three of the hosts.

We first formulated the full model with *hostmin* = 14 and *hostmax* = 15, without any of the S, V and Y constraints. Therefore this model did not prevent any of the guests from meeting more than once. This model had 2640 variables and 866 constraints. The LP objective was 14.0, the LP solution time was 44 s, and branch-and-bound found an integer solution of 14 after 554 s in 1111 nodes. This solution was written out and inspected visually. There were less than 10 violations where guests met more than once and it was possible, by manually adding in the violated constraints, to obtain a feasible solution (with 14 hosts) to the original problem.

We began to suspect that this might be an optimal solution. Therefore we fixed the first 13 boats as hosts and attempted to prove that this was infeasible. We further reduced the size of the model (and relaxed the problem) by restricting the indices of the guest boats in the S, V and Y constraints to 21 ... 42. This model was still large by most standards: 19,470 constraints and 11,664 variables. It was run using a parallel implementation of OSL [3] on seven RS/6000 computers at IBM U.K. Scientific Centre, Hursley, by William Ricketts. The run was aborted after 189 h, having failed to prove infeasibility. OSL had processed about 2500 nodes, of which around 50% were still active: 239 were infeasible. Some nodes were taking over 2 h to evaluate.

9. CONSTRAINT PROGRAMMING APPROACH

This approach was suggested by the desire to prove infeasibility for the 13-host model, in the light of the failure of OSL to find a feasible solution. However, it turned out that Constraint Programming was able to find a feasible solution very rapidly, albeit with manual intervention.

Constraint Programming (CP) has its roots in an area of computer science called constraint solving, dating back to the late 1970s [4]. The algorithms and methods developed in this area have been applied to combinatorial optimization problems, and are widely known in the discipline of Artificial Intelligence, although they are less well known in the Operations Research community. However, in recent years there has been a recognition that CP and IP offer alternative approaches to similar problems, and that a knowledge of both may lead to improved solution of large-scale combinatorial problems. Both approaches essentially require a tree search, and in both the search strategy is guided by an understanding of the structure of the problem.

Combinatorial problems in CP are solved by formulating the model as a Constraint Satisfaction Problem (CSP). A CSP consists of a set of variables, each with a finite set of possible values (its domain), and a set of constraints. Each constraint affects a subset of the variables and restricts the values that those variables can simultaneously take. A solution to a CSP is an assignment of a value to each variable satisfying all the constraints. (See Tsang [5] for an introduction to CSPs and their algorithms).

This problem was formulated as a CSP using ILOG Solver [6], a constraint programming tool in the form of a C++ library. Solver offers a standard backtracking search method, the forward checking algorithm, for solving CSPs, as well as the opportunity (not required in this case) to use special-purpose solution methods. The forward checking algorithm repeatedly chooses an unassigned variable, chooses a value for that variable from its current domain and makes a tentative assignment. The constraints are then used to identify the effects of the assignment on future (still unassigned) variables, and any value in the domain of a future variable which conflicts with the current assignment (and the rest of the current partial solution) is temporarily deleted. If at any stage the domain of a future variable becomes empty, the algorithm backtracks and retracts the last assignment.

As a simple example, consider the following problem: we have three variables A, B and C each with domain $\{1,2,3\}$ and three constraints:

(i) $A + B = 4$,
(ii) $B < C$,
(iii) $A \neq C$.

We use the notation $\mathbf{dom}(A) = \{1, 2, 3\}$. Suppose we start (arbitrarily) by fixing $A = 1$. Then using constraint (i) we can immediately eliminate 1 and 2 from the domain of B, giving $\mathbf{dom}(B) = \{3\}$, but if we then apply constraint (ii) the domain of C becomes empty. Thus we have to backtrack and undo the assignment $A = 1$, and (for the sake of argument) re-assign $A = 2$. By constraint (i) we now have $\mathbf{dom}(B) = \{2\}$ and from constraint (ii) we have $\mathbf{dom}(C) = \{3\}$. Checking with constraint (iii), there is no contradiction and so the algorithm stops with the solution $A = 2, B = 2, C = 3$.

It can be seen that this is an apparently naive enumerative approach which, for larger problems, requires the use of efficient variable and value ordering heuristics for successful implementation; namely how do we choose which variable to fix, and what value do we assign to that variable?

The first advantage of the constraint programming approach was that the formulation as a CSP was much more compact than had been possible as an IP. Since the task was to show (if possible) that the problem with 13 specific host boats, and therefore 29 guest boats, was insoluble, host and guest boats were treated separately in the formulation. Suppose that there are G guest boats, H host boats and T time periods. The main variables, x_{it}, represent the host boat that guest boat i visits at time t; the domain of each x_{it} is the set $\{1, \ldots, H\}$, and there are GT such variables.

The constraints that every guest boat must always have a host are automatically satisfied by any solution to the CSP, which must have a value assigned to each x_{it}.

The constraints that no guest boat can visit a host boat more than once are expressed in the CSP by saying that:

$$x_{i1}, x_{i2}, x_{i3}, \ldots, x_{iT} \text{ are all different for all } i$$

For each i, this is represented by a single CSP constraint, expressed by a single statement in Solver; this gives G constraints.

The capacity constraints are dealt with, as in the IP, by introducing new constrained variables:

$$\begin{aligned} \nu_{ijt} &= 1 \text{ if guest boat } i \text{ visits host } j \text{ at time } t \\ &= 0 \text{ otherwise} \end{aligned}$$

The relationships between these variables and the x_{it} variables are specified by GHT Boolean constraints:

$$\nu_{ijt} = 1 \text{ iff } x_{it} = j \qquad \text{for all } i,j,t$$

and as in the IP, the capacity constraints are then:

$$\sum_{i=1}^{G} c_i \nu_{ikt} \leqslant C_k \qquad \text{for all } k,t$$

where c_i is the crew size of guest boat i and C_i is the spare capacity of host boat k.

The constraints that no pair of crews can meet twice also require the introduction of a new set of variables, representing whether a pair of guest boats k and $l(k < l)$ meet at time t:

$$y_{klt} = 1 \text{ iff crews } k \text{ and } l \text{ meet at time } t$$

The constraints linking the new variables to the original x variables are:

$$\text{if } x_{kt} = x_{lt} \text{ then } y_{klt} = 1 \qquad \text{for all } k,l,t; k < l$$

and the meeting constraints are expressed by:

$$\sum_{t=1}^{T} y_{klt} \leqslant 1 \qquad \text{for all } k,l$$

Because the y variables have only three subscripts, rather than four as in the equivalent set of variables in the IP formulation, the CSP has only $O(n^2t)$ variables and $O(n^2t)$constraints.

In addition to these constraints, which are sufficient to define the problem, a relatively small number of constraints were introduced to reduce the symmetries in the problem, as much as possible. Symmetries can vastly increase the size of the search space and so the amount of work that has to be done; equivalent solutions, which have, for example, two guest boats with the same size crew interchanged throughout, are of no use, and if there are no solutions, searching through many equivalent partial solutions can be extremely time consuming.

First, an ordering was imposed on the time periods (which are otherwise interchangeable); the first guest boat must visit the host boats in order. The second set of constraints distinguishes between guest boats with the same size crew: if (i,j) are such a pair, with $i < j$, then for the first time period, we impose the constraint:

$$x_{i1} \leqslant x_{j1}$$

and, to allow for the fact that both boats may visit the same host at time 1:

$$\text{either } x_{i1} < x_{j1} \text{ or } x_{i2} \leqslant x_{j2}$$

Finally, constraints were added to distinguish (to an extent) between host boats with the same spare capacity: if j and k are two such host boats, with $j < k$, the first guest boat cannot visit host k unless it also visits host j.

10. SOLVING THE PROBLEM

Having set up the problem in Solver, it was next necessary to choose variable and value ordering heuristics, i.e. rules for deciding which variable to consider next and which value to assign to it. Good variable ordering heuristics, in particular, are often crucial to the success of constraint programming methods. A heuristic which is commonly used is based on the "fail-first" principle, that is, choose the variable which is likely to be hardest to assign. In the forward checking algorithm, this means choosing the variable with the smallest remaining domain; ties were broken by choosing the variable involved in most constraints. If there were still ties, variables were considered in the order in which they appeared in the data; since the crew size array was sorted in descending order, this gave priority to the largest crews.

Value ordering heuristics are not so common in constraint programming, partly because there are no good, cheap, generally applicable ones for use with forward checking. Also, if a problem is insoluble, so that a complete search has to be done to prove insolubility, all possible values will eventually have to be considered and the ordering makes no difference. However, a general principle is to choose a value which seems likely to be a successful assignment, and the host boats accordingly were considered in descending order of spare capacity.

The problem formulation was first tested by solving smaller problems than the full 13 hosts, 29 guests, 6 time periods problem. Several smaller problems were solved very quickly (in 1 or 2 s on a SPARCstation IPX), with little backtracking, and the program was shown to be producing correct solutions. However, the full size problems ran for hours without producing any result.

It was then decided to deal with the successive time periods separately within the program. A complete assignment for one time period was produced before going on to consider the next, but the solution for each time period constrained the solutions for future time periods. In effect, this was another variable ordering heuristic, taking priority over the others, except that the program was not able to backtrack to earlier time periods. The plan was to find a solution for as many time periods as could be solved within a short time, and to print out the domains of the remaining variables at that point. It was hoped that this would give some clue as to why the program could not find a solution for the following time period. With this modification, the program in fact found a solution for 5 time periods very quickly (which it had not previously been able to do). At this point, the domain of any future variable contains those hosts that the corresponding boat can visit and has not already visited. An attempt was made to fit the guest crews into those host boats which they could still visit, by hand, in order to see why it could not be done. However, a solution was found which appeared to be feasible; adding some of the crucial assignments to the program as extra constraints confirmed that a solution based on these assignments did obey all the constraints. Hence, the 13 hosts, 29

Table 4. An optimal solution

	*T*1	*T*2	*T*3	*T*4	*T*5	*T*6
Host						
1	34,35	30	29,31	14,25	32,40	20,36
2	31,32,33	17,41	14,20,40	24,29	36,37	15,34
3	14,39	20,22,31	16,23	28,38	19,24,35	26,30,42
4	22,28	25,39	24,38	15,16	23,29,42	14,21,37
5	18,38	27,29,33	22,41	39	16,20	28,31
6	19,21	16,34	25,28	17,33,42	39	22,35
7	23,24	15,38	30,37	18,32,41	14,22	16,33
8	16,17	26,28,32	35,39	19,22	21,38	24,27
9	27,30	19,37	21,42	26,31, 34,40	15,28	38,41
10	37,42	14,23,36	19,27	21,30	26,33,41	17,18, 25,40
11	20,25, 26,29	24,42	15,17,32	27,35	18,30,34	19,23
12	15,41	21,40	18,26,36	20,23	17,27	39
13	36,40	18,35	33,34	37	25,31	29,32

guests, 6 time periods problem, which had been thought to be insoluble, had been solved, though with some manual assistance. This solution is given in Table 4.

Subsequently, the additional constraints on the 6th time period were removed, and the program allowed to search for a solution without this intervention; it found a solution for the 6th time period in 27 min, and went on to find a solution for the 7th time period in another minute, so that the party could have lasted for longer without requiring more hosts!

The heuristic of considering each time period successively clearly has its limitations; the program declared that there was no solution for the 8th time period based on the solutions to the previous 7, but this does not, of course, mean that there is no solution for 8 time periods with the 13 specified host boats. Sometimes considering all time periods together gives better results than separating them; a much smaller problem with 15 boats in all had been proved using IP to require 5 hosts for a 4-period party. This was confirmed using the Solver program, and it was shown that the corresponding 5-period problem was insoluble, whereas considering the time periods separately did not yield a solution for 4 time periods.

11. DISCUSSION

One of the pleasing aspects of this problem was the way in which two solution techniques complemented each other in solving it rapidly. IP and CP and alone were unable to solve the problem optimally, but together they provided a very fast method of finding an optimal solution. The fundamental difficulty with IP appears to lie in finding a compact representation of the problem. The major stumbling-block is the problem of modelling constraints of the form "all the values in the set $\{x_1 \dots x_n\}$ are distinct" without an exponential number of zero–one variables. However, constraint programming requires far fewer constraints and variables to represent the problem. This is possible because of the greater expressive power of the constraints allowed in constraint satisfaction problems, which are not restricted to linear inequalities as in linear programming. In turn, the greater expressiveness is allowed by the fact that constraint satisfaction algorithms make relatively unsophisticated (although very effective) use of the constraints.

The role of heuristics in finding a solution is clearly crucial; for instance, CP was unable to find a solution with 13 hosts when all the time periods were considered together. Considering each time period in turn has obvious limitations; because backtracking to earlier time periods is not allowed, it is not guaranteed to find a solution if there is one, and it cannot show that a problem is infeasible. Ironically, it seems very probable that if the 13-host problem had indeed been infeasible, the constraint programming approach would not have been able to prove infeasibility: although it is easy to show that a solution with 12 hosts is impossible, because the capacity constraints cannot be met even for a single time period, a problem which is "only just" infeasible, because the meeting and capacity constraints cannot be simultaneously satisfied for the required number of time periods, would be extremely difficult to prove infeasible.

This is a difficult problem. It does not belong to a known category of combinatorial optimization problem, although it does have some similarities to the school timetabling problem, where teachers have to be assigned to classes and time periods. This problem has extra constraints. It can be applied to the assignment of students for group projects, where each group has to be supervised by a member of staff, and a number of group projects are carried out over the course of the academic year. If we require that any two students can work together at most once, and that each member of staff cannot supervise the same student twice, this can be seen to be the same problem. Here the capacity constraints are imposed by the ability of the individual staff members to cope with a given number of students at once.

The model could conceivably also be applied to logistics problems where there is the possibility of contamination of materials at different processing sites, or in military or security problems where there is a need to prevent collusion between (say) inspectors of nuclear installations.

Acknowledgements—We are very grateful to William Ricketts of IBM U.K. Scientific Centre, Hursley Park, Winchester, for his enthusiastic help with the computational experiments with the large model. We are also grateful to Gareth Jones of the Faculty of Mathematical Studies, University of Southampton, for his help with the theoretical results, and to an anonymous reviewer for suggesting another application of the model.

REFERENCES

1. XPRESSMP (Version 7), Dash Associates, Blisworth House, Blisworth, Northants NN7 3BX (1994).
2. H. P. Williams, The elimination of integer variables, *J. Op Res. Soc.* **43**, 387–393 (1992).
3. Optimization Subroutine Library (OSL), IBM Corporation.
4. A. Mackworth, Consistency in networks of relations. *Artific. Intelligence* **8**, 99–118 (1977).
5. E. Tsang, *Foundations of Constraint Satisfaction*. Academic Press (1993).
6. J.-F. Puget, A C++ implementation of CLP. In *Proceedings of SPICIS94* (1994).

Readers may address inquiries to Dr S. C. Brailsford at: scb@maths.soton.ac.uk