

On Solving The Progressive Party Problem as a MIP

Erwin Kalvelagen
erwin@gams.com
GAMS Development Corp.
Washington DC

19th June 2001

Abstract

The ‘Progressive Party Problem’ [9] has long been considered a problem intractable for branch-and-bound mixed integer solvers. Quite impressive results have been reported with constraint programming systems for this problem. As a result the problem has become a standard example in texts on constraint programming. Fortunately, there has been progress in the mixed integer programming arena: we can solve now larger and more difficult problems than ever before. Improvements in algorithmic theory, solvers, modeling environments and computer hardware created a new situation, where reported cases of unsolvable instances of MIP models need to be re-examined, possibly with another outcome. In this paper we will show that we can solve the ‘Progressive Party Problem’ formulated as a large MIP problem, using standard, off-the-shelf hardware and software. A simple myopic heuristic is also implemented and can solve the problem in a fraction of the time.

1 Introduction

The ‘Progressive Party Problem’ was first stated by Peter Hubbard, a member of the Sea Wych Owners Association, and of the Mathematics Department of Southampton University [12]. Consider an evening party during a yachting rally. There are n boats and their crews. A number of boats is to be chosen as host boat: they will host a party where other crews will visit in time slots $t = 1..T$ of half an hour. The total number of time periods T is given. Guest crews will go from one host boat to another. Host boats have a given capacity: the number of guests they can receive for a party. Guest crews can not visit the same host boat more than once and guest crews can not meet other guest crews more than once. Find a schedule that minimizes the number of host boats.

The following data is available: $T = 6$, $n = 42$ and the crew sizes and guest capacities are listed in table 1.

The real-world problem had a few extra conditions: the first three boats are designated hosts boats: the organizer and two crews consisting of parents need to stay at their boats to be reachable. The last few boats, with zero capacity, are virtual boats: they are “crews” consisting of children that visit parties but can not host a party.

The problem has been reported to be unsolvable using Mixed Integer Programming [9, 7]. There has been success using Constraint Programming techniques or mixed Constraint Programming/Integer Programming systems. [9] uses the ILOG Solver with some manual intervention, [10] can solve some instances (including the full-blown problem) using Oz, [7] solves some smaller instances using a MLLP (Mixed Logical-Linear Programming) solver. [5] uses Constraint Programming and Local Search for models with even more time periods and [8] uses a combined strategy using Eclipse and Cplex.

2 Model formulation

In this section we will review some of the formulations used in earlier unsuccessful attempts to solve the model as a mixed integer programming problem.

The first formulations are due to [9]. We introduce binary variables $x_{i,j,t}$,

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

Table 1: Crew sizes and guest capacities

h_i defined as follows:

$$x_{i,j,t} = \begin{cases} 1 & \text{if crew } j \text{ visits boat } i \text{ at time slot } t, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

$$h_i = \begin{cases} 1 & \text{if boat } i \text{ is a host boat,} \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Let the data be denoted by w_i and p_i as being the crew size and the guest capacity of boat i .

The objective is to minimize the number of host boats:

$$\min z = \sum_i h_i \quad (3)$$

Parties only take place at hosts boats, so if $h_i = 0$ then we must have $x_{i,j,t} = 0$:

$$x_{i,j,t} \leq h_i \quad \forall i \neq j, t \quad (4)$$

The guest capacity of a host boat can not be exceeded:

$$g_i := \max\{p_i - w_i, 0\} \quad (5)$$

$$\sum_{j|j \neq i} w_j x_{i,j,t} \leq g_i \quad \forall i, t \quad (6)$$

where g_i is a parameter indicating the maximum number of guests a host boat can accomodate.

Crews are not supposed to be idle at any time: they are either visiting a host boat or they serve as host crew:

$$h_j + \sum_{i|i \neq j} x_{i,j,t} = 1 \quad \forall j, t \quad (7)$$

The constraint that crews cannot meet more than once is split into two parts: first a guest crew cannot visit a host crew more than once and second, between guest crews the meeting limit is imposed.

Crews can not visit the same boat more than once:

$$\sum_t x_{i,j,t} \leq 1 \quad \forall i \neq j \quad (8)$$

Guest crews can not meet more than once:

$$\sum_{(i,t)|i \neq j, i \neq j'} x_{i,j,t} x_{i,j',t} \leq 1 \quad \forall j \neq j' \quad (9)$$

The last equation is nonlinear and can be rewritten as a set a linear constraints. Several suggestions can be found in the literature.

$$x_{i,j,t} + x_{i,j',t} + x_{i',j,t} + x_{i',j',t} \leq 3 \quad \begin{array}{l} \forall (i, i', j, j', t, t') | \\ i \neq j, i \neq j', i' \neq j, i' \neq j', \\ i \neq i', j < j', t \neq t' \end{array} \quad (10)$$

The number of equations this creates is however prohibitively large [9]. A different formulation (also from [9]) introduces extra binary variables $y_{i,j,j',t}$ defined by

$$y_{i,j,j',t} = \begin{cases} 1 & \text{if crews } j \text{ and } j' \text{ visit boat } i \text{ at time slot } t, \\ 0 & \text{otherwise.} \end{cases} \quad (11)$$

and the following constraints are formulated to implement this:

$$y_{i,j,j',t} \leq \frac{x_{i,j,t} + x_{i,j',t}}{2} \quad \forall (i, j, j', t) | i \neq j, i \neq j', j < j' \quad (12)$$

$$y_{i,j,j',t} \geq x_{i,j,t} + x_{i,j',t} - 1 \quad \forall (i, j, j', t) | i \neq j, i \neq j', j < j' \quad (13)$$

$$\sum_{(j',t) | j < j'} y_{i,j,j',t} \leq 1 \quad \forall i, j \quad (14)$$

As the host boat i is irrelevant in this respect, the following formulation is more appropriate [10]. Define binary variables $m_{j,j',t}$ instead of $y_{i,j,j',t}$ and assume only that $m_{j,j',t} = 1$ if crews j and j' meet at a party at time t . If they don't meet then, we leave $m_{j,j',t}$ unrestricted. The linking equation reduces to:

$$m_{j,j',t} \geq x_{i,j,t} + x_{i,j',t} - 1 \quad \forall (j, j', t) | j < j' \quad (15)$$

$$\sum_t m_{j,j',t} \leq 1 \quad \forall j < j' \quad (16)$$

Disappointing results are reported by [9]. Commercial LP/MIP solvers XPRESS and OSL were tried unsuccessfully, even on smaller instances and relaxations. A footnote in [10] mentions that the improved formulation using equations (15) and (16) was tried by the authors of [9], without apparent success.

A somewhat different formulation is found in [7]. First of all, variables of the form $x_{i,i,t}$ are explicitly included in the model. These variables are set to one for host crews by including a constraint:

$$x_{i,i,t} \geq h_i \quad \forall i, t \quad (17)$$

We think it is better not to include these variables at all in the model. I.e. in our implementation, $x_{i,j,t}$ is only defined for $i \neq j$. This reduces the number of integer variables and the above constraint is not needed.

A more distinctive feature is the usage of general integer variables $v_{j,t}$ defined by:

$$v_{j,t} = \text{the host boat visited by crew } j \text{ at period } t \quad \forall j, t \quad (18)$$

where $v_{i,t} = i$ for host crews. The implication

$$v_{j,t} = i \Rightarrow x_{i,j,t} = 1 \quad (19)$$

is modeled as:

$$x_{i,j,t} \geq 1 - \alpha_{i,j,t} - \beta_{i,j,t} \quad \forall i, j, t \quad (20)$$

$$i - v_{j,t} \geq 1 - M(1 - \alpha_{i,j,t}) \quad \forall i, j, t \quad (21)$$

$$v_{j,t} - i \geq 1 - M(1 - \beta_{i,j,t}) \quad \forall i, j, t \quad (22)$$

using additional binary variables $\alpha_{i,j,t}$ and $\beta_{i,j,t}$. For this big- M formulation, M can be chosen as $M = n$. An alternative formulation is:

$$v_{j,t} = \sum_i i x_{i,j,t} \quad \forall j, t \quad (23)$$

The variables $m_{j,j',t}$ are calculated using $v_{j,t}$ in a similar fashion as just described. The implication

$$v_{j,t} = v_{j',t} \Rightarrow m_{j,j',t} = 1 \quad (24)$$

is implemented as:

$$h_j + h_{j'} + m_{j,j',t} \geq 1 - \phi_{j,j',t} - \psi_{j,j',t} \quad \forall t, j < j' \quad (25)$$

$$v_{j',t} - v_{j,t} \geq 1 - M(1 - \phi_{j,j',t}) \quad \forall t, j < j' \quad (26)$$

$$v_{j,t} - v_{j',t} \geq 1 - M(1 - \psi_{j,j',t}) \quad \forall t, j < j' \quad (27)$$

using additional binary variables $\phi_{j,j',t}$ and $\psi_{j,j',t}$. Again M can be chosen as $M = n$. It is observed that $v_{j,t}$ can be declared as continuous variables as they will automatically assume integral values.

The authors [7] could not solve the full instance of this formulation using the Cplex solver. For $n = 10$, computations were terminated after 20000 nodes.

3 Solving the model

The basic model we will use is discussed in the first part of previous section:

$$\begin{aligned}
\min z &= \sum h_i \\
x_{i,j,t} &\leq h_i \\
\sum_j w_j x_{i,j,t} &\leq g_i \\
h_j + \sum_i x_{i,j,t} &= 1 \\
\sum_t x_{i,j,t} &\leq 1 \\
m_{j,j',t} &\geq x_{i,j,t} + x_{i,j',t} - 1 \\
\sum_t m_{j,j',t} &\leq 1
\end{aligned}$$

The model from [7] with its big- M formulations and general integer variables, looked less attractive as a starting point.

A few changes have been applied to this model.

The variables $m_{j,j',t}$ can be relaxed to continuous variables, as they assume integer values automatically, that is at least the ones that are restricted. The unrestricted ones being possibly fractional are not an issue.

We fixed the designated hosts boats:

$$h_i = 1 \quad i = 1, 2, 3 \quad (28)$$

The virtual boats can be fixed to being guest boats:

$$h_i = 0 \quad i = 40, 41, 42 \quad (29)$$

This can be slightly generalized: any boat with a net guest capacity smaller than the smallest crew size will never be a hoat boat. I.e.

$$\bar{w} := \min_i w_i \quad (30)$$

$$h_i = 0 \quad i | g_i < \bar{w} \quad (31)$$

It can be argued that there is a bound

$$z \geq 13 \quad (32)$$

as we need 13 host boats just to get a valid schedule for a one period party. This can easily be verified just by running the model for $T = 1$. The resulting model is:

$$\begin{aligned} \min z &= \sum h_i \\ x_{i,j} &\leq h_i \\ \sum_j w_j x_{i,j} &\leq g_i \\ h_j + \sum_i x_{i,j} &= 1 \end{aligned}$$

In fact, as mentioned by [3] quite a simplified model can be used to obtain this bound. A simplified version of that is:

$$\begin{aligned} \min z &= \sum h_i \\ \sum_i w_i &\leq \sum_i p_i h_i \end{aligned}$$

This model gives $z^* = 13$ implying the bound on z for the full-blown model. The same bound can be derived by sorting the boats on capacity and choosing the first k boats that can accomodate all crews, which yields $k = 13$.

Another observation is that we replace equation (6) by:

$$\sum_{j|j \neq i} w_j x_{i,j,t} \leq p_i h_i \quad \forall i, t \quad (33)$$

which may be tighter if h_i is still fractional. Similarly, we can replace equation (8) by:

$$\sum_t x_{i,j,t} \leq h_i \quad \forall i \neq j \quad (34)$$

As we were interested in finding a solution with $z = 13$ we fixed the objective variable to this value and told Cplex to place emphasis on feasibility. In addition we set branching priorities as follows:

- First pay attention to h_i , then worry about $x_{i,j,t}$
- First handle large crews, then do smaller crews

Surprisingly the primal simplex much faster for this model compared to the default dual simplex method, not only for the root node but also for the subsequent nodes.

Using these options we were able to solve the model. The results are reproduced in table 2.

number of equations	220060
number of variables	15541
number of binary variables	10368
number of nonzero elements	678997
gams model compilation time	0 s.
gams model generation time	4 s.
cplex total solution time	9054 s.
cplex relaxed lp solution time	165 s.
cplex total iterations	777860
number of nodes	507

Table 2: Result for the full model

The results are running GAMS and GAMS/CPLEX 7.0 on a PC with a 1.2 GHz AMD Athlon processor, 512 MB Ram running Linux. Interestingly, the same model solves even faster on Windows ME on the same machine: 1375 seconds, 107541 iterations, 197 nodes. The complete model is listed in Appendix A.

4 A time staged heuristic

[9] mentions how they could find a solution using the ILOG Solver Constraint Programming System. They first solve the system for $t = 1$, then fixed variables and solved for $t = 2$ etc. Interestingly they did not try the same scheme using a MIP solver. In this section we will show results based on the following heuristic: first solve for $t = 1$, then fix h_i and $x_{i,j,1}$. Then solve for $t = 1, 2$. Now we can fix $x_{i,j,2}$. Solve for $t = 1, 2, 3$ etc. Note that we can not fix the variables $m_{j,j',t}$ as they are partially left undefined. As was done in [9], we were even able to find a solution for a seventh time period without increasing the number of host boats.

The complete model that implements this, is reproduced in Appendix B. The basic results are given in 3.

time stage	rows	cols	nz	disc. vars	obj	gen. time	sol. time
1	38830	2620	114130	1758	13	0.73	1.62
2	73270	3445	146371	1722	13	1.17	1.72
3	107710	4306	181672	1722	13	1.58	2.35
4	142150	5167	216973	1722	13	2.10	3.00
5	176590	6028	252274	1722	13	2.52	3.84
6	211030	6889	287575	1722	13	2.96	4.39
(7)	245470	7750	322876	1722	13	3.42	5.50

Table 3: Result for the time staged model

rows	number of equations
cols	number of variables
nz	number of nonzero elements
disc. vars	number of discrete variables
obj.	optimal value of objective variable
gen. time	gams generation time (seconds)
sol. time	cplex solution time (seconds)

Table 4: Explanation of headers in table 3

The models grow in size but are very easy to solve. The total turn around time is less than a minute.

As the optimal objective solution for stage 1 through 7 remains at 13, we know this is an optimal solution. However, this strategy is not guaranteed to work: we were lucky that fixing the earlier stages did not cause the objective to deteriorate or that the subproblems became infeasible.

5 Discussion

The fact that we can solve models of this size on our desktops can be attributed to the following factors. First, there has been tremendous improve-

ments in hardware. The desktop PC that was used to run this model is quite a garden variety machine. However, it offers unprecedented computing power. [4] mentions that a 1.2 GHz AMD Athlon can outperform a Cray C90 (16 procs, 4.2 ns), a mighty machine just a decade ago, yielding 558 Mflop/s compared to 479 Mflop/s for the C90 on a standard Linpack benchmark. (This comparison even favors the Cray somewhat as that machine was designed to do dense linear algebra very well). This is truly amazing, especially considering that these PC's, most of the time, are sitting idle or running mail readers, web browsers or word processing software.

Another important reason being able to solve this model is the huge progress that has been made in the MIP solvers. Both the capabilities in solving large LP's reliably and the incorporation of new theoretic results from research on integer programming have lead to new classes of models that can now be solved routinely. A good exposé on this aspect can be found in [2]. As an example we mention the presolver, a feature that nowadays is a standard part of most commercial solvers. For our large model, the performance of the presolver is summarized in table 5. The resulting model is still very large, but the presolver is surely a very good job on this model.

	before	after
rows	220060	125085
cols	15541	11928
nz	678997	396392

Table 5: Model size reduction by the presolver

The last important tool to use is a modeling system like AMPL[1] or GAMS[6]. The use of a modeling system almost invites the user to experiment with alternative formulations. Especially for very large models, modeling systems provide a concise representation that can be understood in full. Opposed to (computer) programming, where large problems can be decomposed into smaller ones in a natural fashion (e.g. by stepwise refinement[13] etc.) modeling is characterized by “simultaneous equations”. A compact notation is then called for to maintain a proper “helicopter view” on the model. Integer models especially are very sensitive to different formulations. Often a different formulation can make the difference between being able to solve a model or not. This requires easy experimentation, with running different

variations of a model. Modeling systems are very suited in such an environment, as there is virtual no barrier between the idea of a reformulation and its implementation. Often it is a question of minutes after the inception of a possible new idea to try out, and the actual run being started. Unfortunately with MIP models, reformulations are very dependent on the particular model and data. For this big model, that means that, although you can try out new ideas on smaller instances, ultimately one will need to evaluate reformulations on the real full-blown model. From the results in table 2 we see that the overhead of using a modeling system is quite small, even compared to solving the relaxed LP. For large models we see that generation times are going up linearly with the number of nonzero elements, while solution times certainly don't. When writing little algorithms like in the time staged model, the overhead can be larger due to regeneration of similar models, a property GAMS currently does not take advantage of.

References

- [1] <http://www.ampl.com>
- [2] R. E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, R. Wunderling, *MIP: Theory and Practice Closing the Gap*, System Modelling and Optimization: Methods, Theory and Applications, Kluwer, The Netherlands, M. J. D. Powell and S. Scholtes, editors, pp. 19-49, 2000.
- [3] S. C. Brailsford, P. M. Hubbard, B. M. Smith and H. P. Williams, *Organizing a Social Event – A Difficult Problem of Combinatorial Optimization*, Computers and Operations Research, 23, pp. 845–856, 1996.
- [4] J. J. Dongarra, *Performance of Various Computers Using Standard Linear Equations Software*, Report CS-89-85, University of Tennessee, January 2001.
- [5] P. Galinier and J.K. Hao, *Solving the progressive party problem by local search*, Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization, S. Voss, S. Martello, I.H. Osman and C. Roucairol (Eds.), Kluwer Academic Publishers, Chapter 29, pp. 418–432, 1998.
- [6] <http://www.gams.com>

- [7] J. N. Hooker and M. H. Osorio, *Mixed Logical-Linear Programming*, Discrete Applied Mathematics, 96-97, pp. 395–442, 1999.
- [8] Rodosek R., Wallace M. G. and Hajian M. T., *A New Approach to Integrating Mixed Integer Programming with Constraint Logic Programming*, in Annals of Operational Research, Recent Advances in Combinatorial Optimization, Vol 86, pp. 63–87, 1999.
- [9] B. M. Smith, S. C. Brailsford, P. M. Hubbard and H. P. Williams, *The Progressive Party Problem: Integer Linear Programming and Constraint Programming Compared*, Report 95.8, University of Leeds, School of Computer Studies, March 1995. (Also appeared in *Proceedings of First International Conference on Principles and Practice of Constraint Programming (CP'95)*, Springer Verlag LNCS 976, pp. 36–52, Cassis, September 1995 and in *Constraints*, vol. 1, pp. 119–138, 1996).
- [10] J. P. Walser, *Solving Linear Pseudo-Boolean Constraint Problems with Local Search*, Proceedings of the 14th National Conference on Artificial Intelligence, AAAI-97, Providence, RI, 1997.
- [11] J. P. Walser, *Domain-Independent Local Search for Linear Integer Optimization*, Ph.D. dissertation, Universität des Saarlandes, 1998.
- [12] J. P. Walser, <http://www.ps.uni-sb.de/~walser/ppp/ppp.html>.
- [13] N. Wirth, *Program Development by Stepwise Refinement*, CACM, v. 14, n. 4, April 1971, pp. 221–227.

Appendix A: Full model

```
$title Progressive Party Problem -- Full blown model
$ontext
```

```
    This model solves the 6 period model.
```

```
    Erwin Kalvelagen, june 2001
```

```
$offtext
```

```
set i 'boat or crew number' /b1*b42/;
```

```

parameter capacity(i) 'max number of guests' /
    b1  6, b2  8, b3 12, b4 12, b5 12, b6 12, b7 12, b8 10,
    b9 10, b10 10, b11 10, b12 10, b13 8, b14 8, b15 8, b16 12,
    b17 8, b18 8, b19 8, b20 8, b21 8, b22 8, b23 7, b24 7,
    b25 7, b26 7, b27 7, b28 7, b29 6, b30 6, b31 6, b32 6,
    b33 6, b34 6, b35 6, b36 6, b37 6, b38 6, b39 9, b40 0,
    b41 0, b42 0
/;

parameter crew(i) 'crew size' /
    b1  2, b2  2, b3  2, b4  2, b5  4, b6  4, b7  4, b8  1,
    b9  2, b10 2, b11 2, b12 3, b13 4, b14 2, b15 3, b16 6,
    b17 2, b18 2, b19 4, b20 2, b21 4, b22 5, b23 4, b24 4,
    b25 2, b26 2, b27 4, b28 5, b29 2, b30 4, b31 2, b32 2,
    b33 2, b34 2, b35 2, b36 2, b37 4, b38 5, b39 7, b40 2,
    b41 3, b42 4
/;

parameter guest_cap(i) 'guest_capacity';
guest_cap(i) = max(capacity(i)-crew(i),0);

set t 'time slot' /t1*t6/;

alias (i,j,ii,jj);
alias (t,tt);

set nd(i,j) 'off-diagonal';
nd(i,j)$(ord(i)<>ord(j)) = yes;

set lti(i,j) 'less-than';
lti(i,j)$(ord(i)<ord(j)) = yes;

variables
    nh                'number of host boats'
    x(i,j,t)          'crew j visits party i at time slot t'
    h(i)               'boat i is a host boat'
    meet(j,jj,t)      'crews j and jj meet at time t'
;
binary variables x,h;

equations
    obj                'objective'
    host(i,j,t)        'parties only on host boats'
    cap(i,t)           'capacity constraint'
    crewhost(j,t)      'crew is a host or is hosted (no idle crews)'

```

```

        visitonce(i,j) 'crew can only visit a boat once'
        link(i,j,jj,t) 'calculates meet(j,jj)'
        meetonce(j,jj) 'any pair of guest crews can meet only once'
;

*
* minimize number of host boats
*
obj.. nh =e= sum(i, h(i));

*
* there are only parties on host boats
*
host(nd(i,j),t).. x(i,j,t) =l= h(i);

*
* max number of guest that a host boat can handle
*
cap(i,t).. sum(nd(i,j), crew(j)*x(i,j,t)) =l= guest_cap(i)*h(i);

*
* no idle crews: a crew is either hosting a party, or visiting
* a party
*
crewhost(j,t).. h(j) + sum(nd(i,j), x(i,j,t)) =e= 1;

*
* max one visit to each host boat
*
visitonce(nd(i,j)).. sum(t, x(i,j,t)) =l= h(i);

*
* guest crews can meet only once
* with aid of extra binary variables
*
meet.lo(lti(j,jj), t) = 0;
meet.up(lti(j,jj), t) = 1;
link(i,lti(j,jj),t)$ (nd(i,j) and nd(i,jj))..
    meet(j,jj,t) =g= x(i,j,t) + x(i,jj,t) - 1;

meetonce(lti(j,jj)).. sum(t, meet(j,jj,t)) =l= 1;

*
* some boats are designated host boats
*

```

```

set must_be_host(i) /b1,b2,b3/;
h.fx(must_be_host) = 1;

*
* some boats are designated guest boats
* (the virtual boats).
*
set must_be_guest(i) /b40,b41,b42/;
h.fx(must_be_guest) = 0;

*
* make sure boats with very limited guest capacity are never
* selected as host boats
* (this will include the virtual boats, so what)
*
scalar mincrew 'smallest crew';
mincrew = smin(j, crew(j));
h.fx(i)$(guest_cap(i) < mincrew) = 0;

*
* it can be shows that 12 host boats is not enough
* to handle even a single period. Try to find a schedule
* with 13 boats.
*
nh.fx = 13;

model m /all/;

*
* priorities
* -----
* first branch on h(i), sorted on crew size
* then branch on x(i,j,t), sorted on crew size
*
parameter mx 'max crew size';
mx = smax(i,crew(i));
h.prior(i) = mx - crew(i);
x.prior(i,j,t) = mx + sqr(card(i)) - ord(i) - ord(j);
m.prioropt=1;

* solve to optimality
option optcr=0;

* increase iteration and time limit
option iterlim=1000000;

```

```

option reslim=100000;

option mip=cplex;

*
* cplex option file
*
file f /cplex.opt/;
putclose f 'heurfreq -1''mipinterval 1''startalg 1''subalg 1''mipemphasis 1''/;
m.optfile=1;

solve m using mip minimizing nh;

*
* sanity check
*
parameter meetcount(j,jj);
meetcount(nd(j,jj)) = sum((i,t), x.l(i,j,t)*x.l(i,jj,t));
abort$sum((j,jj)$ (meetcount(j,jj) > 1.5),1) "meeting condition is not met";

```

Appendix B: Time staged model

```

$title Progressive Party Problem -- Time Staged Approach
$ontext

```

This model find a 7 period schedule using a time staged heuristic.

Erwin Kalvelagen, 2001

```

$offtext

```

```

set i 'boat or crew number' /b1*b42/;

```

```

parameter capacity(i) 'max number of guests' /
    b1  6,  b2  8,  b3 12,  b4 12,  b5 12,  b6 12,  b7 12,  b8 10,
    b9 10,  b10 10, b11 10, b12 10, b13 8,  b14 8,  b15 8,  b16 12,
    b17 8,  b18 8,  b19 8,  b20 8,  b21 8,  b22 8,  b23 7,  b24 7,
    b25 7,  b26 7,  b27 7,  b28 7,  b29 6,  b30 6,  b31 6,  b32 6,
    b33 6,  b34 6,  b35 6,  b36 6,  b37 6,  b38 6,  b39 9,  b40 0,
    b41 0,  b42 0
/;

```

```

parameter crew(i) 'crew size' /
    b1  2, b2  2, b3  2, b4  2, b5  4, b6  4, b7  4, b8  1,
    b9  2, b10 2, b11 2, b12 3, b13 4, b14 2, b15 3, b16 6,
    b17 2, b18 2, b19 4, b20 2, b21 4, b22 5, b23 4, b24 4,
    b25 2, b26 2, b27 4, b28 5, b29 2, b30 4, b31 2, b32 2,
    b33 2, b34 2, b35 2, b36 2, b37 4, b38 5, b39 7, b40 2,
    b41 3, b42 4
/;

parameter guest_cap(i) 'guest_capacity';
guest_cap(i) = max(capacity(i)-crew(i),0);

set t 'time slot' /t1*t7/;

alias (i,j,ii,jj);

set nd(i,j) 'off-diagonal';
nd(i,j)$(ord(i)<>ord(j)) = yes;

set lti(i,j) 'less-than';
lti(i,j)$(ord(i)<ord(j)) = yes;

variables
    nh                'number of host boats'
    x(i,j,t)          'crew j visits party i at time slot t'
    h(i)              'boat i is a host boat'
    meet(j,jj,t)      'crews j and jj meet at time t'
;
binary variables x,h;

equations
    obj                'objective'
    host(i,j,t)        'parties only on host boats'
    cap(i,t)           'capacity constraint'
    crewhost(j,t)      'crew is a host or is hosted (no idle crews)'
    visitonce(i,j)     'crew can only visit a boat once'
    meetonce(j,jj)     'any pair of guest crews can meet only once'
    link(i,j,jj,t)    'calculates meet(j,jj)'
;

set td(t) 'dynamic set';

*

```

```

* minimize number of host boats
*
obj.. nh =e= sum(i, h(i));

*
* there are only parties on host boats
*
host(nd(i,j),td).. x(i,j,td) =l= h(i);

*
* max number of guest that a host boat can handle
*
cap(i,td).. sum(nd(i,j), crew(j)*x(i,j,td)) =l= guest_cap(i)*h(i);

*
* no idle crews: a crew is either hosting a party, or visiting
* a party
*
crewhost(j,td).. h(j) + sum(nd(i,j), x(i,j,td)) =e= 1;

*
* max one visit to each host boat
*
visitonce(nd(i,j)).. sum(td, x(i,j,td)) =l= h(i);

*
* guest crews can meet only once
* with aid of extra binary variables
*
meet.lo(lti(j,jj), t) = 0;
meet.up(lti(j,jj), t) = 1;
link(i,lti(j,jj),td)$ (nd(i,j) and nd(i,jj))..
    meet(j,jj,td) =g= x(i,j,td) + x(i,jj,td) - 1;

meetonce(lti(j,jj)).. sum(td, meet(j,jj,td)) =l= 1;

*
* some boats are designated host boats
*
set must_be_host(i) /b1,b2,b3/;
h.fx(must_be_host) = 1;

*
* some boats are designated guest boats
* (the virtual boats).

```

```

*
set must_be_guest(i) /b40,b41,b42/;
h.fx(must_be_guest) = 0;

*
* make sure boats with very limited guest capacity are never
* selected as host boats
* (this will include the virtual boats, so what)
*
scalar mincrew 'smallest crew';
mincrew = smin(j, crew(j));
h.fx(i)$(guest_cap(i) < mincrew) = 0;

*
* it can be shown that 12 host boats is not enough
* to handle even a single period
*
nh.lo = 13;

model m /obj, host, cap, crewhost, visitonce, link, meetonce/;

* solve to optimality
option optcr=0;

* don't generate fixed variables
m.holdfixed = 1;

* keep listing file small
option limrow = 0;
option limcol = 0;
m.solprint = 0;

option mip=cplex;

loop(t,

*
* add new member to dynamic set
*
    td(t) = yes;

    solve m using mip minimizing nh;
    abort$(m.modelstat <> 1) "model became infeasible";

*

```

```

* fix variables for this time stage
*
  h.fx(i)$(ord(t)=1) = h.l(i);
  x.fx(i,j,t) = x.l(i,j,t);
);

display h.l,x.l;

*
* sanity check
*
parameter meetcount(j,jj);
meetcount(nd(j,jj)) = sum((i,t), x.l(i,j,t)*x.l(i,jj,t));
abort$sum((j,jj)$(meetcount(j,jj) > 1.5),1) "meeting condition is not met";

```