

Progress on the Progressive Party Problem

Helmut Simonis*

Cork Constraint Computation Centre
University College Cork
h.simonis@4c.ucc.ie

We report new results for solving the progressive party problem with finite domain constraints, which are competitive with the best local search results in the literature. When introduced in [1], the progressive party problem was a show case for finite domain constraint programming, since a solution of the original problem for 6 time periods could be found in 26 minutes with Ilog Solver, while an integer programming model was not successful. Improved results using finite domains were reported in [2]. Since then, alternative solutions using MILP have been proposed, while local search methods have been significantly faster, as well as more stable for given problem variations. The best results (to my knowledge) are from [3].

Details of the problem can be found on its CSPLIB page (<http://www.csplib.org>). The problem consists of assigning guest teams of different sizes to host boats (different capacities) for multiple team periods. Each guest can visit a host atmost once, and two guest teams should also meet atmost once. We look for feasible solutions for different numbers of time periods and/or host/guest sizes.

We have explored two models for the problem. The first considers the complete problem by posting all variables for all time periods together. It creates `alldifferent` constraints for all variables of each guest team, expressing that a guest team can visit a host atmost once. For each time period we state a `bin-packing` constraint [4] to handle the capacity constraints for the host boats. A collection of reified equality constraints between pairs of guests handle the condition that two guests meet atmost once.

When analysing the behaviour of the first model, it was easy to see that there was very limited interaction between time periods, and no effective constraint propagation for one level until variables of that level were assigned. This suggested a decomposition model (already used in [1]) which sets up the problem for one time period at a time. If guests are assigned to the same host in a time period, then disequality constraints are imposed between them for future time periods. Each layer adds new disequality constraints, so that the problem for each following layer becomes more constrained. These disequalities can be expressed with a global `some-different` [5] constraint, although we use the binary decomposition. The model then consists of a single `bin-packing` constraint and a `some-different` constraint expressing all disequalities between the variables for a time period. Surprisingly, we do not loose propagation with this decomposed model compared to the first model, while speeding up execution significantly.

The key to solving this problem is a customized search routine, which combines five elements. We to solve the problem period by period, i.e. assigning all variables

* This work is supported by Science Foundation Ireland (Grant Number 05/IN/I886) and by Cisco Systems Inc. through the Cisco Collaborative Research Initiative.

for one period before starting on the next. This provides a more stable method than using `first_fail` over all variables. We use a `first_fail` variable selection inside each time period. For each time period we use of a partial search strategy (credit-based search [2]) to avoid deep backtracking in dead parts of the search tree. A randomized value selection leads to a more equitable choice of values. Once this randomization element is in place, we can also use a restart strategy which cuts off search after all credit in some layer has been expended, and begins exploration from the top again.

We compare results obtained with our second model in ECLiPSe 6.0 with the results published in [3]. Except for a single problem instance (4/9), results are comparable. A full version of the paper can be found at <http://4c.ucc.ie/~hsimonis/party.pdf>

Table 1. Results on Problems from [3]

Problem	Size	ECLiPSe 6.0					Comet				
		Solved	Min	Max	Avg	σ	Solved	Min	Max	Avg	σ
1	6	100	0.187	0.343	0.226	0.027	100	0.33	0.38	0.35	0.01
1	7	100	0.218	0.515	0.271	0.044	100	0.39	0.49	0.44	0.02
1	8	100	0.250	2.469	0.382	0.258	100	0.50	0.72	0.57	0.04
1	9	100	0.266	9.906	1.253	1.734	100	0.74	1.46	1.01	0.15
1	10	100	0.375	136.828	23.314	21.738	100	1.47	41.72	4.68	5.80
2	6	100	0.218	2.375	0.624	0.484	100	0.37	0.52	0.43	0.04
2	7	100	0.266	3.453	1.117	0.873	100	0.47	1.64	0.73	0.18
2	8	100	0.297	15.421	2.348	2.263	100	0.75	7.16	2.69	1.26
2	9	100	0.469	107.187	20.719	21.162	99	4.41	162.96	33.54	34.10
3	6	100	0.219	3.266	0.551	0.504	100	0.37	0.56	0.43	0.04
3	7	100	0.250	3.734	0.889	0.705	100	0.49	1.45	0.74	0.18
3	8	100	0.296	21.360	2.005	2.417	100	0.84	11.64	2.85	1.55
3	9	100	1.078	173.266	34.774	32.731	96	4.41	164.44	40.10	40.14
4	6	100	0.219	9.922	2.443	2.146	100	0.39	0.72	0.47	0.05
4	7	100	0.360	25.297	3.531	3.421	100	0.55	2.33	0.87	0.26
4	8	100	0.438	53.547	8.848	9.193	100	1.23	11.38	3.68	1.91
4	9	63	3.062	494.109	206.324	161.250	94	8.35	166.90	59.55	44.44
5	6	100	0.203	7.922	1.498	1.441	100	0.53	5.29	1.67	0.75
5	7	100	0.266	28.000	5.889	5.463	100	1.77	132.82	29.72	28.76
6	6	100	0.219	15.219	2.147	2.661	100	0.58	31.84	2.74	3.31
6	7	100	0.407	64.312	11.328	12.290	88	3.24	152.37	56.92	48.91

References

1. Brailsford, S., Hubbard, P., Smith, B., Williams, H.: Organizing a social event - a difficult problem of combinatorial optimization. *Computers Ops Res* **23** (1996) 845–856
2. Beldiceanu, N., Bourreau, E., Chan, P., Rivreau, D.: Partial search strategy in CHIP. In: 2nd International Conference on Metaheuristics MIC97, Sophia Antipolis, France (1997)
3. Van Hentenryck, P., Michel, L.: *Constraint-Based Local Search*. MIT Press, Boston (2005)
4. Shaw, P.: A constraint for bin packing. In: *Principles and Practice of Constraint Programming - CP 2004*, Toronto, Canada (2004) 648–662
5. Richter, Y., Freund, A., Naveh, Y.: Generalizing alldifferent: The somedifferent constraint. In: *Principles and Practice of Constraint Programming - CP 2006*, Nantes, France (2006) 468–483