

Improved Models for Graceful Graphs

Jean-François Puget¹ and Barbara M. Smith²

¹ ILOG, 9 avenue de Verdun, 94250 Gentilly, France
puget@ilog.fr

² Cork Constraint Computation Centre, University College Cork, Ireland
b.smith@4c.ucc.ie

Abstract. The problem of finding a graceful labelling of a graph, or proving that the graph is not graceful, has previously been modelled as a CSP. A new and much faster CSP model of the problem has been recently proposed. This model was inspired by a mathematical proof. We present a third model that is in some sense a combination of the previous two, but is much more efficient than either. We give several new results for graphs whose gracefulness was previously unknown. The possibility to generalize the use of the modelling techniques presented here is discussed.

1 Introduction

The problem of finding graceful labelling of a graph has been investigated in the CP community because these are good test beds for symmetry breaking methods, and because several open problems seem to be easily solved using constraint programming.

A labelling f of the nodes of a graph with q edges is *graceful* if f assigns each node a unique label from $\{0, 1, \dots, q\}$ and when each edge xy is labelled with $|f(x) - f(y)|$, the edge labels are all different. (Hence, the edge labels are a permutation of $1, 2, \dots, q$.)

Gallian [4] gives a survey of graceful graphs, i.e. graphs with a graceful labelling, and lists the graphs whose status is known; his survey is frequently updated to include new results. Graceful labelling were first defined by Rosa in 1967, although the name was introduced by Golomb [6] in 1972. Gallian lists a number of applications of labelled graphs; however, the study of graceful graphs has become an active area of research in its own right. The survey lists several classes of graph for which it is known that every instance is graceful, for instance the wheel graph W_n , consisting of a cycle C_n and an additional node joined to every node of the cycle. However, the only general result given by Gallian is that if every node has even degree and the number of edges is congruent to 1 or 2 (mod 4) then the graph is not graceful. For example, the cycles C_{4n+1} and C_{4n+2} are not graceful, although C_{4n} and C_{4n+3} are graceful for all n . There is a long-standing conjecture that all trees are graceful and although this has been proved for several classes of tree (including paths), and for all trees with at most 27 nodes, the general case remains unproven.

Given a graph whose gracefulness is so far unknown, in general there is no way to tell whether it is graceful or not, except by trying to label it. Constraint programming is thus a useful tool to use in investigating graceful graphs.

2 A Direct Model

A simple CSP model was introduced in [9]. It has two sets of variables: a variable for each node, x_0, x_1, \dots, x_{n-1} each with domain $\{0, 1, \dots, q\}$ and a variable for each edge, d_0, d_1, \dots, d_{q-1} , each with domain $\{1, 2, \dots, q\}$. The value assigned to x_i is the label attached to node i , and the value of d_k is the label attached to the edge k .

The constraints of the problem are: if edge k joins nodes i and j then $d_k = |x_i - x_j|$, for $k = 1, 2, \dots, q$; x_0, x_1, \dots, x_{n-1} are all different; d_0, d_1, \dots, d_{q-1} are all different (and form a permutation).

Let us consider graph $K_3 \times P_2$ depicted in Fig. 1. There are 6 variables x_i corresponding to the vertices of the graph, and 9 variables d_j corresponding to the edges.

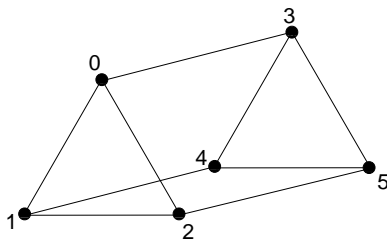


Fig. 1. The graph $K_3 \times P_2$.

There are two kinds of symmetry in the problem of finding a graceful labelling of a graph: first, there may be symmetry in the graph. For instance, if the graph is a clique, any permutation of the node labels in a graceful labelling is also graceful, and if the graph is a path, P_n , the node labels can be reversed. The second type of symmetry is that we can replace the value v of every node variable x_i by its complement $q - v$. We can also combine each graph symmetry with the complement symmetry. For instance, the graceful labelling $(0, 3, 1, 2)$ of P_4 has three symmetric equivalents: reversal $(2, 1, 3, 0)$; complement $(3, 0, 2, 1)$; and reversal + complement $(1, 2, 0, 3)$.

If the graph is symmetric, it is essential to eliminate all or most of the symmetry in the CSP in order to avoid wasted search, especially if all graceful labellings are required or the graph is not graceful. In terms of the CSP model described above, the symmetry of the graph leads to variable symmetries in the CSP affecting the node variables x_1, x_2, \dots, x_n ; each symmetry maps an assignment of a value l to a variable x_i to an assignment of the same value to a variable x_j . Following the procedure introduced by Crawford *et al.* [3], variable symmetries can be eliminated by adding a lexicographic ordering constraint for each symmetry. Although in general this may not be practicable, we showed in [11] that when there is an allDifferent constraint on the variables, the symmetry can be eliminated by at most $n - 1$ binary constraints.

The symmetries of the graph in Figure 1 are, firstly, any permutation of the 3-cliques which acts on both in the same way, for instance, transposing nodes 1 and 2 and simultaneously nodes 4 and 5. Secondly, the labels of the first clique (nodes 1, 2, 3) can be interchanged with the labels of the corresponding nodes (nodes 4, 5, 6) in the second.

A possible set of constraints to eliminate the graph symmetry is $x_1 < x_2$, $x_2 < x_3$, to exclude permutations within the cliques and $x_1 < x_4$, $x_1 < x_5$, $x_1 < x_6$ to exclude swapping the cliques and permuting both. Since the constraints imply that $x_1 = 0$, this constraint together with $x_2 < x_3$ is sufficient.

We have shown in [13] that combinations of value symmetries with variable symmetries could be broken by a combination of lexicographic ordering constraints and element constraints. We state one such constraint for each combination of the complement symmetry with one variable symmetry.

The model just described, which will be called the *node model* in the rest of the paper, is adequate to find all graceful labellings of the graph in Figure 1, and allowed us to investigate slightly larger graphs as well, e.g. the related graphs $K_4 \times P_2$, $K_5 \times P_2$, and $K_6 \times P_2$. $K_5 \times P_2$ is graceful, and has a unique graceful labelling, apart from symmetric equivalents, whereas $K_6 \times P_2$ is not graceful; these new results are now listed in Gallian's survey [4]. Results that we obtained with this model for other classes of graph are summarized in [16]. However, the search effort and run time increase rapidly with problem size, so that it is very time-consuming, as shown below, to prove that $K_7 \times P_2$, with 14 nodes and 49 edges, is not graceful; $K_8 \times P_2$ is out of reach within a reasonable time. In the next sections, a better model of the problem is introduced.

3 The Edge-Label Model

A more efficient model has been recently introduced in [17]. We summarize it here. Its roots are in a search algorithm by Beutner & Harborth [2]. They use a backtracking search procedure, based on the approach in the proof, to find graceful labellings or prove that there are none. They focus on the edge labels, considering each label in turn, from q (the number of edges) downwards. Each edge label has to be used, and appear somewhere in the graph, and they construct a labelling, or prove that there is no labelling, by trying to decide, for each edge label in turn, which node labels will appear at either end of the edge. The classes they consider are nearly complete graphs, with specific kinds of sub graph removed from a clique; for instance, they show that $K_n - e$, i.e. an n -clique with one edge removed, is graceful only for $n \leq 5$ and that any graph derived from K_n by removing 2 or 3 edges is not graceful for $n > 6$.

The edge-label model has a variable e_i for each edge label i , which indicates how this edge label will be formed. The value assigned to e_i is the smaller node label forming the edge labelled i , i.e. if $e_i = j$, the edge labelled i joins the nodes labelled j and $j + i$. Hence, the domain of e_i is $\{0, 1, \dots, q - i\}$. (Essentially, an edge label is associated with a *pair* of node labels, but since if the edge label and the smaller node label are known, the other node label is also known, we need only associate an edge label with one node label.) Note that the domain of e_q is $\{0\}$. The domain of e_{q-1} is initially $\{0, 1\}$ but this can be reduced either to $\{1\}$ or to $\{0\}$ arbitrarily, to break the complement symmetry. We choose to set $e_{q-1} = 1$ (although choosing $e_{q-1} = 0$ instead makes little difference).

There is also a variable l_j for each node label j ; the value of l_j represents the node to which the label j is attached, provided that this node label is used at all in the labelling. To allow for the fact that a label may not be used, the domains of these variables contain

a dummy value, say $n+1$. The node label variables must all have different values (unless they are assigned the dummy value).

The link between the two sets of variables is achieved by the constraints: $e_i = j$ iff the values of l_j and l_{j+i} represent adjacent nodes in the graph ($1 \leq i \leq q$; $0 \leq j \leq q - i$). In Solver 6.3, this is expressed as a table constraint, using a ternary predicate which uses the adjacency matrix of the graph. GAC is enforced on table constraints, which is relatively slow.

The graph symmetry, if any, has still to be dealt with. In the edge-label model, it appears as value symmetries affecting the node-label variables. If a graph symmetry σ maps the node numbered i to the node numbered $\sigma(i)$, then it maps the assignment $l_j = i$ to the assignment $l_j = \sigma(i)$, whereas in the node model, it maps $x_i = j$ into $x_{\sigma(i)} = j$. Law and Lee [8] point out that value symmetry can be transformed into variable symmetry acting on the dual variables and then is amenable to the Crawford *et al.* procedure for deriving symmetry-breaking constraints.

Hence, a simple way to derive symmetry-breaking constraints in the edge-label model is to re-introduce the variables x_1, \dots, x_n from the node model. These are the duals of the node label variables l_1, \dots, l_q . The two sets of variables can be linked by the channelling constraints: $x_i = j$ iff $l_j = i$ for $i = 1, \dots, n$ and $j = 0, \dots, q$, or equivalently by the global *inverse* constraint [1]. As a side-effect, the channelling constraints are sufficient to ensure that every possible node label is assigned to a different node, or else is not used, so that no explicit *allDifferent* constraint is required.

It was shown in [17] that another symmetry breaking technique, SBDS [5] is more efficient for several classes of graphs. We found that dynamic lex constraints (DLC) [14], yet another symmetry breaking technique, is even more efficient than SBDS on these problems. Therefore, we will use DLC for breaking symmetries in the edge-label model.

The search strategy considers the edge labels in descending order; for each one in turn, the search first decides which node labels will make that edge label, and then decides where in the graph to put those node labels. Hence, both the edge label variables e_1, e_2, \dots, e_q and the node label variables l_0, l_1, \dots, l_q are search variables. The variable ordering strategy finds the largest edge label i that has not yet been attached to a specific edge in the graph. The next variable assigned is e_i , if that has not been assigned; if e_i has been assigned a value j , then the next variable is l_j , if that is not yet assigned, or l_{i+j} . (If all three variables have been assigned, the label i is already associated with a specific edge.)

Since the domains of the edge label variables increase in size as the labels decrease, at least initially, this variable ordering is approximately equivalent to minimum domain ordering, as far as the edge label variables are concerned. However, limited experiments with choosing the edge label variable with smallest domain, if different from the largest unassigned edge label variable, did not give good results. As in the node model, the value ordering chooses the smallest available value in the domain.

4 A Combined Model

The efficiency of the edge-label model is due in part to its clever search order. The search focuses first on the edges labelled with the largest values. A similar idea can be adapted to the node model: we can try to label first the edge with the largest value. One simple way to do this is to introduce a variable y_j for each edge label, whose value will be the edge labelled with j . These variables are the duals of the node label variables d_0, \dots, d_q in the original model. The two sets of variables can be linked by the channelling constraints: $d_i = j$ iff $y_j = i$ for $i = 0, \dots, n$ and $j = 0, \dots, q$, or equivalently by the global *inverse* constraint [1].

Therefore, we consider the following model. It has three sets of variables: a variable for each node, x_0, x_1, \dots, x_{n-1} each with domain $\{0, 1, \dots, q\}$, a variable for each edge, d_0, d_1, \dots, d_{q-1} , each with domain $\{1, 2, \dots, q\}$, and a variable for each edge label y_1, y_2, \dots, y_q . The value assigned to x_i is the label attached to node i , and the value of d_k is the label attached to the edge k .

The constraints of the problem are: if edge k joins nodes i and j then $d_k = |x_i - x_j|$, for $k = 1, 2, \dots, q$; x_1, x_2, \dots, x_n are all different; d_1, d_2, \dots, d_q are all different (and form a permutation); d_i and y_j are linked by an inverse constraint.

The graph symmetries, if any, need to be dealt with. Any graph symmetry induces a value symmetry for the variables y_j . If edge i is mapped to edge k by a symmetry, then the assignment $y_j = i$ is mapped to $y_j = k$. These value symmetries can be broken using dynamic lex constraints as explained in [14]. Therefore we use DLC to break these symmetries.

The complement symmetry does not impact edge labels. Therefore, this symmetry is not broken by the DLC method on the y_j variables. The complement symmetry induces a value symmetry for the x_i variables, exactly as in the original node model. We can break it using a lexicographic constraint and an element constraint as in [13].

The search strategy is similar to that used in the edge-label model. It considers the edge labels in descending order; for each one in turn, the search first decides which node labels will make that edge label, and then decides where in the graph to put those node labels. Hence, both the edge label variables y_1, y_2, \dots, y_q and the node label variables x_0, x_1, \dots, x_{n-1} are search variables. The variable ordering strategy finds the largest edge label i that has not yet been attached to a specific edge in the graph. The next variable assigned is y_i , if that has not been assigned; if y_i has been assigned a value j then we look at both ends of edge j . Suppose that these are nodes a and b . Then both variables x_a and x_b are eligible as the next search variable. When there are several node variables eligible as the next search variable, the one corresponding to the largest edge label is selected first.

As in the node model and in the edge-label model, the value ordering chooses the smallest available value in the domain.

5 Experimental Results

All three models have been implemented with ILOG Solver 6.3 [7]. Experiments were run on a 1.4GHz Pentium M laptop computer. Graph symmetries are computed auto-

matically using the AUTOM routine [12]. Running time includes symmetry detection time.

Several classes of graphs were considered. The class $K_n \times P_2$ was studied in [9], [10], and [17] among others. These graphs are made of two cliques of size n plus n edges linking pairs of nodes, one from each clique. The graph in Figure 1 is the smallest graph in this class. We also considered graphs where the number of cliques is larger than 2. We report for each graph the number of solutions (0 when the graph has no graceful labelling), and for each method the number of backtracks and the running time. Results are given in table 1. It was not previously known that $K_5 \times P_3$ is graceful and that $K_4 \times P_4$ is not.

Table 1. Results for $K_n \times P_m$ graphs.

Graph	Solutions	Node model		Edge model		Combined model	
		BT	sec.	BT	sec.	BT	sec.
$K_3 \times P_2$	4	46	0	14	0.01	15	0.01
$K_4 \times P_2$	15	908	0.2	151	0.32	166	0.03
$K_5 \times P_2$	1	22182	5.48	725	5.26	956	0.31
$K_6 \times P_2$	0	544549	396	1559	41	2538	1.09
$K_7 \times P_2$	0			1986	140.8	4080	3.13
$K_8 \times P_2$	0			2041	325.8	4620	6.58
$K_9 \times P_2$	0			2045	668.8	4690	11.7
$K_3 \times P_3$	284	5363	0.48	1833	1.88	1939	0.33
$K_4 \times P_3$	704	1248935	195	155693	883	183150	25.1
$K_5 \times P_3$	101					7049003	1974
$K_3 \times P_4$	12754	1936585	205	370802	964	424573	37.2
$K_4 \times P_4$	0					416756282	76635

The results for the $K_n \times P_2$ class clearly show a significant improvement in running times when going from the node model to the edge-label model, and yet another significant improvement when going from the edge-label model to the combined model. However, the edge-label model remains the one that leads to the smallest search tree. When the number of cliques is larger than 2, the node model is faster than the edge-label model, but again the latter model gives the smallest search trees. The combined model is by far the most efficient of the three, in terms of run-time.

We considered a related class, $K_n \times C_m$. Graphs in this class are made of m cliques of size n , where corresponding nodes in each cliques are connected in a cycle of length m . Results are given in table 2. The graphs $K_3 \times C_3$ and $K_5 \times C_3$ are not graceful because of the parity condition given earlier (all nodes are of even degree and the number of edges is $\equiv 1$ or $2 \pmod{4}$). However, the results that there is no graceful labelling for the graphs $K_6 \times C_3$, $K_3 \times C_4$, and $K_4 \times C_4$ are new.

These results show again that the combined model is by far the most efficient one. For these graphs, the size of the search tree for the edge-label model is comparable to that for the combined model. The run-time for the node model and the edge-label model are comparable.

Table 2. Results for $K_n \times C_m$ graphs.

Graph	Solutions	Node model		Edge model		Combined model	
		BT	sec.	BT	sec.	BT	sec.
$K_3 \times C_3$	0	5059	0.74	604	1.08	599	0.14
$K_4 \times C_3$	22	3104352	665	47532	522	51058	10.8
$K_5 \times C_3$	0					1184296	509
$K_6 \times C_3$	0					12019060	38431
$K_3 \times C_4$	0					206244	24.1
$K_4 \times C_4$	0					169287859	40233

We now consider another class of graphs, double wheel graphs: these consist of a pair of cycles C_n , with each node joined to a common hub. Previous results with the node model are given in [16]. Results for all three models are given in Table 3.

Table 3. Results for double wheel graphs.

Graph	Solutions	Node model		Edge model		Combined model	
		BT	sec.	BT	sec.	BT	sec.
DW3	0	198	0.03	20	0.03	21	0.01
DW4	44	4014	0.54	378	0.6	408	0.11
DW5	1216	132632	16.9	14140	24.7	14565	1.62
DW6	35877	6740234	1028	891351	2572	916253	102

For this class of graphs, the combined model is again the most efficient. The tree size is comparable for the edge-label model and the combined model. For these graphs, the node model is faster than the edge-label model. This is probably because these graphs are very far from complete graphs, and indeed quite sparse, whereas the edge-label model was derived from an algorithm tailored for almost complete graphs. This explanation may hold for the graphs $K_n \times P_m$ with $m > 2$ as well. Surprisingly enough, the combined model is very efficient on these sparse graphs.

The last class of graphs was studied in [17]. These are dense graphs, consisting of m cliques of size n that share a common clique of size r . They are denoted $B(n, r, m)$. Results for this class are given in table 4.

We can summarize our experimental results as follows. The combined model is by far the most efficient model, whereas the edge-label model is the one that leads to the smallest search trees. For dense graphs, the edge-label model is much more efficient than the node model, but for sparse graphs, the reverse is true.

6 Discussion

We have presented a new model for graceful graphs. It is derived from the classical node model and the recently proposed edge-label model. It reuses the variables and

Table 4. Results for multi-clique graphs.

Graph	Solutions	Node model		Edge model		Combined model	
		BT	sec.	BT	sec.	BT	sec.
B(3,2,2)	4	7	0.01	1	0.01	1	0
B(4,2,2)	4	127	0.03	20	0.04	26	0
B(5,2,2)	1	2552	0.45	75	0.32	88	0.04
B(6,2,2)	0	48783	18.9	109	2.21	135	0.15
B(7,2,2)	0			120	7.46	161	0.42
B(8,2,2)	0			120	23.4	163	6.22
B(6,3,2)	0	20698	4.7	102	1.48	141	0.12
B(6,4,2)	0	5102	1.21	47	0.53	77	0.05
B(6,5,2)	0	470	0.26	11	0.12	16	0.02
B(5,3,3)	5	10109	1.88	266	1.76	346	0.15
B(6,3,3)	0	996667	591	1588	49.7	1924	1.18
B(7,3,2)	0	49763	314	135	6.35	193	0.34

the constraints of the former, and the search strategy of the latter. The resulting model is quite simple, yet much more efficient than both its predecessors. We were able to derive new mathematical results using that model, namely we have shown that $K_5 \times P_3$ is graceful, and found all its graceful labellings, while several other graphs have been shown to have no graceful labelling. These results in themselves are interesting enough. Moreover, some of modelling techniques we describe here could be useful in other domains. These techniques are the following ones.

The first one is to make more use of duality. Instead of using the original variables (the edge variables d_i in the node model) as search variables, we use their duals (the y_j variables in the combined model). This amounts to selecting first which value to assign to the d_i variables, and then selecting which variable d_i will receive that value. As we show, using the dual variables in this way can improve the performance of the model even when they play no other role. This technique can be applied to any permutation problem; for instance, Régim used a similar idea in solving sports scheduling problems [15].

The second technique is to group variables together in the search. When an edge variable d_i is assigned a value, search proceeds with the variables x_a and x_b corresponding to both ends of the edge. These variables are linked by the constraint $d_i = |x_a - x_b|$. Therefore, the second technique amounts to following the constraint graph during search.

The last technique is a quite general one. Both the edge-label model and the combined model use a clever search strategy. We believe that the search strategy should often be considered as part of the model. Indeed, we tried various other variable ordering strategies, such as smallest domain, most constrained variable, etc., and none of them gave good results. The search strategy that we used is crucial to the efficiency of these models.

Acknowledgments

The second author is supported by the Science Foundation Ireland under Grant No. 00/PI.1/C075.

References

1. N. Beldiceanu. Global constraints as graph properties on structured network of elementary constraints of the same type. Technical Report Technical Report T2000/01, SICS, 2000.
2. D. Beutner and H. Harborth. Graceful labelings of nearly complete graphs. *Results in Mathematics*, 41:34–39, 2002.
3. J. Crawford, M. Ginsberg, E. Luks, and A. Roy. Symmetry-Breaking Predicates for Search Problems. In *Proceedings KR'96*, pages 149–159, 1996.
4. J. A. Gallian. A Dynamic Survey of Graph Labeling. *The Electronic Journal of Combinatorics*, (DS6), 2005. 9th edition.
5. I. P. Gent and B. M. Smith. Symmetry Breaking During Search in Constraint Programming. In W. Horn, editor, *Proceedings ECAI'2000, the European Conference on Artificial Intelligence*, pages 599–603, 2000.
6. S. W. Golomb. How to number a graph. In R. C. Read, editor, *Graph Theory and Computing*, pages 23–37. Academic Press, 1972.
7. ILOG. *ILOG Solver 6.3. User Manual* ILOG, S.A., Gentilly, France, July 2006.
8. Y. C. Law and J. H. M. Lee. Symmetry Breaking Constraints for Value Symmetries in Constraint Satisfaction. *Constraints*, 11:221–267, 2006.
9. I. J. Lustig and J.-F. Puget. Program Does Not Equal Program: Constraint Programming and Its Relationship to Mathematical Programming. *INTERFACES*, 31(6):29–53, 2001.
10. K. E. Petrie and B. M. Smith. Symmetry Breaking in Graceful Graphs. Technical Report APES-56-2003, APES Research Group, 2003. Available from <http://www.dcs.st-and.ac.uk/~apes/apesreports.html>.
11. J.-F. Puget. Breaking symmetries in all different problems. In *Proceedings of IJCAI'05*, pages 272–277, 2005.
12. J.-F. Puget. Automatic detection of variable and value symmetries. In van Beek, P., ed., *Principles and Practice of Constraint Programming - CP 2005*, LNCS 3709, pages 475–489. Springer, 2005.
13. J.-F. Puget. An Efficient Way of Breaking Value Symmetries. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, pages 117–122. AAAI Press, 2006.
14. J.-F. Puget. Dynamic Lex Constraints. In Benhamou, F., ed., *Principles and Practice of Constraint Programming - CP 2006*, LNCS 4204, pages 453–467. Springer, 2006.
15. J.-C. Régim. Constraint Programming and Sports Scheduling Problems. *Inform*, May 1999, Cincinnati.
16. B. Smith and K. Petrie. Graceful Graphs: Results from Constraint Programming. <http://4c.ucc.ie/~bms/Graceful/>, 2003.
17. B.M. Smith: Constraint Programming Models for Graceful Graphs. In Benhamou, F., ed., *Principles and Practice of Constraint Programming - CP 2006*, LNCS 4204, pages 545–559. Springer, 2006.