Modelling Langford's Problem: A Viewpoint for Search

Özgür Akgün and Ian Miguel

School of Computer Science, University of St Andrews
{ozgur.akgun,ijm}@st-andrews.ac.uk

Abstract. The performance of enumerating all solutions to an instance of Langford's Problem is sensitive to the model and the search strategy. In this paper we compare the performance of a large variety of models, all derived from two base viewpoints. We empirically show that a channelled model with a static branching order on one of the viewpoints offers the best performance out of all the options we consider. Surprisingly, one of the base models proves very effective for propagation, while the other provides an effective means of stating a static search order.

1 Introduction

Langford's Problem (number 24 at www.csplib.org) is specified as follows.

> Arrange k sets of numbers 1 to n in a sequence, so that each appearance of the number m is m numbers on from the last. For example, the L(3,9) problem is to arrange 3 sets of the numbers 1 to 9 so that the first two 1's and the second two 1's appear one number apart, the first two 2's and the second two 2's appear two numbers apart, etc.

Alternative constraint programming models and search strategies for this problem were explored first by Smith [17]. Smith presented two models: the first corresponds to what we call the *Positional* approach in this paper (Section 2.2), and the second model is the dual of the first one. When viewed as a permutation problem, the dual viewpoint for a permutation may be constructed by interchanging the variables and the values. This approach has been studied further and generalised by [20,18,12,11,16]. Smith's second model corresponds to the *Direct* approach in this paper (Section 2.1) with one exception. In our direct approach, a value m is repeated k times, whereas in [17] the second occurrence of a value m is represented with $m + n$, the third with $m + 2n$, and so on.

In this paper we investigate the performance of a number of different models and search strategies for Langford's Problem. We first present two approaches to modelling Langford's Problem, using two separate viewpoints. We then employ a very strict channelling constraint between the two viewpoints to construct a channelled approach. We compare 20 variations in total, including the best search heuristic identified by [17] and the AllSAT solver `BC_MINISAT_ALL` [19] to enumerate all solutions. Our empirical evaluation shows that a static branching order on the Direct viewpoint in a channelled model offers the best performance.

2 Two Approaches to Modelling Langford's Problem

In what follows we present two approaches to modelling this problem, using two separate viewpoints, in the ESSENCE [7,8,9] constraint specification language. One is direct, has a sequence of numbers just like the problem specification. The other is positional, has a function from numbers and the repetition to the position in the sequence.

2.1 A *Direct* Approach

We first present a *Direct* problem specification for Langford's Problem, given in Figure 1. An ESSENCE specification identifies: the input parameters of the problem class (`given`), whose values define an instance; the combinatorial objects to be found (`find`); the constraints the objects must satisfy (`such that`); identifiers declared (`letting`); and an (optional) objective function (`min/maximising`). This specification contains a single decision variable with a domain of type `sequence`, with a fixed length. In order to solve an ESSENCE specification, it is first refined into a solver-independent constraint model using the CONJURE automated constraint modelling system [1,2,3,4] and then prepared for input to a particular constraint solver, such as MINION [10], or SAT by the SAVILE ROW system [13,14,15].

Sequences in ESSENCE are not required to be of fixed length in general. A fixed length sequence is very similar to a one-dimensional array, except for the provision of some additional operators. For example, in this specification we use the `preImage` operator (on line 29) to denote all the index positions of a certain number. The apartness constraint (on lines 11–21) is written as a triply-nested quantified expression. For each number there exists a starting position: the first index where we see this particular number in the sequence. Once this position is denoted, the apartness constraint can be posted by constraining the appropriate positions in the sequence to be equal to the number in focus. As an alternative to the existential quantification we could have used auxiliary variables to denote the first positions of each number as a separate ESSENCE variable. However, in this case we feel the existential quantification is clearer. Furthermore, our constraint modelling pipeline of CONJURE and SAVILE ROW is able to generate a top level decision variable for the existentially quantified variables automatically. The second constraint (on line 24) is for breaking the reflection symmetry. The third constraint (lines 26–29) is implied: it states the fact that each number must occur k times in the sequence.

2.2 A *Positional* Approach

A second specification of Langford's Problem is what we call the *Positional* approach, presented in Figure 2. This contains a single decision variable with a domain of type function. The function maps 2-tuples, where the first component of the tuple is a number between 1 and n and the second component is the repetition index. Each 2-tuple is mapped to a position in the sequence and since

```
1  language Essence 1.3
2
3  given k : int(2..)
4  given n : int(1..)
5
6  letting seqLength be k * n
7
8  $ The sequence of numbers
9  find seq : sequence (size seqLength) of int(1..n)
10
11 $ The apartness constraint
12 such that
13  $ for each number
14  forAll number : int(1..n) .
15   $ there exists a starting position
16   $ (i.e. the first position where number occurs)
17   exists start : int(1..seqLength) .
18    $ the following positions all contain the same value
19    $ start, start+(number+1), start+2*(number+1), ...
20    forAll i : int(1..k) .
21     seq(start + (i-1) * (number+1)) = number
22
23 $ symmetry breaking
24 such that seq(1) < seq(seqLength)
25
26 $ Each number from 1 to n appear exactly k times in seq.
27 $ This is an implied constraint.
28 such that
29  forAll i : int(1..n) . |preImage(seq, i)| = k
```

Fig. 1. ESSENCE Specification: *Direct* Approach to Modelling Langford's Problem.

these positions need to be distinct, the function is marked to be injective. CONJURE produces an all-different constraint when refining this injective function. The apartness constraint (on lines 14–21) is written as a doubly-nested quantified expression, and for each number repetition pair it posts a binary constraint. The second constraint (on line 27) is for breaking the reflection symmetry.

2.3 Channelling the Direct and Positional Approaches

We combine the direct and positional into a single channelled [6] approach. In order to do so we concatenate the two specifications into one file, excluding the common definitions like the k and the n. We keep all the constraints regarding the problem definition in both individual specifications. The two symmetry breaking constraints are not compatible with each other, so if we post both of them we would lose solutions. We keep one or the other in our empirical evaluation.

```
1  language Essence 1.3
2
3  given k : int(2..)
4  given n : int(1..)
5
6  letting number be domain int(1..n)
7  letting repetition be domain int(1..k)
8  letting position be domain int(1..k*n)
9
10 $ The positions of all repetitions of all numbers
11 find pos : function (total, injective)
12                    tuple (number, repetition) --> position
13
14 $ Occurrences of number i must be i+1 places apart.
15 $ So if the number 4 appears at position 3,
16 $ the next occurrence of it must be at position 8,
17 $ leaving a gap of 4 positions.
18 such that
19  forAll i : number .
20   forAll j : int(2..k) .
21    pos(tuple (i,j)) = pos(tuple (i,j-1)) + i + 1
22
23 $ symmetry breaking
24 $ The first occurrence of the number 1 is closer to the
       beginning than its last occurrence is to the end.
25 such that
26  pos(tuple (1,1)) - 1 < k*n - pos(tuple (1,k))
```

Fig. 2. ESSENCE Specification: *Positional* Approach to Modelling Langford's Problem.

The channelling constraints (Figure 3) are very tight by design. Specifically, they only allow one assignment to the `seq` variable for each assignment to the `pos` variable and vice versa. We validate this by removing the problem constraint from either viewpoint and enumerating all solutions.

The first channelling constraint (on lines 1–5) set values of `seq` from values of `pos`. The second channelling constraint (on lines 8–11) does the inverse. The third constraint (on lines 13–17) is not required when both problem constraints are present, but when the Positional constraints are removed they provide tighter channelling and break symmetry.

3 Empirical Evaluation

We compare the performance of solving 80 instances using the two base approaches, variations of channelled approach and a number of search strategies. The instances are generated for values of 2..6 for k, and for values of 2..17 for

```
1  $ from pos to seq
2  such that
3   forAll i : number .
4    forAll j : repetition .
5     seq(pos(tuple (i,j))) = i
6
7  $ from seq to pos
8  such that
9   forAll i : int(1..k*n) .
10   exists j : repetition .
11    pos(tuple (seq(i),j)) = i
12
13 $ entries in pos are ordered
14 such that
15  forAll i : number .
16   forAll j : int(2..k) .
17    pos(tuple (i,j-1)) < pos(tuple (i,j))
```

Fig. 3. Channelling constraints for integrating the *Direct* and *Positional* models in Essence

n. We use a timeout of four hours for each model-instance pair. Three instances $((2,15),(2,16),(2,17))$ cannot be solved with any of our models, and these are excluded from our analysis. A further thirty instances are trivial (solved by one of the base models in under 5 search nodes), and these are kept in the plots. We analyse the 47 non-trivial instances in more detail. The Essence specifications, the parameter files, scripts for rerunning the experiments, and the raw data containing our results can be found in a source code repository hosted at http://github.com/stacs-cp/ModRef2018-Langfords.

3.1 Comparing the Two Base Approaches

We compare the Direct and Positional approaches, using our modelling pipeline (Conjure 2.2.0, Savile Row 1.7.0) to produce constraint models suitable for input to Minion 1.8. We employ the default static branching order of Minion: the default variable ordering is by order of appearance, and the default value ordering is lexicographic. In general the positional model performs better (Figure 4): it is able to enumerate all solutions to all but three of out 77 instances. The Direct model times out for 40 instances and performs much worse on the small number of instances it can solve.

A more advanced variable ordering heuristic is weighted-degree [5]. Minion implements two variations of weighted-degree heuristics, the plain `wdeg` and another one that also takes the domain size into account: `domoverwdeg`. In Figure 5, we see that the Positional model performs even better with one of the weighted degree heuristics (in comparison to static ordering) and it is significantly better than the Direct model.

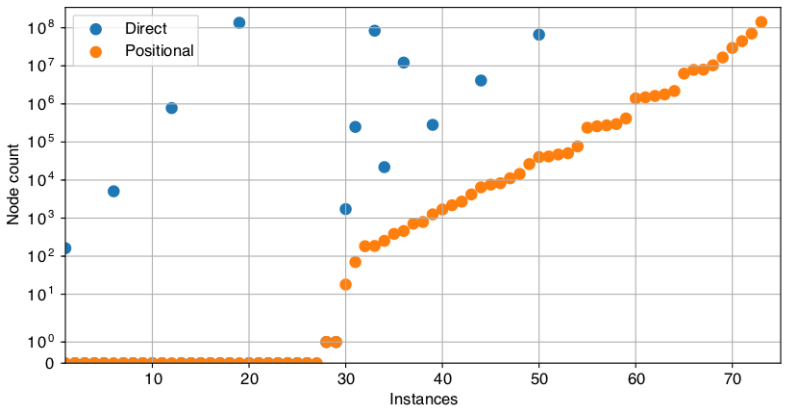

Fig. 4. Search nodes with static variable ordering with the Direct model vs the Positional model. Sorted by Positional. The timed-out instances are not plotted.

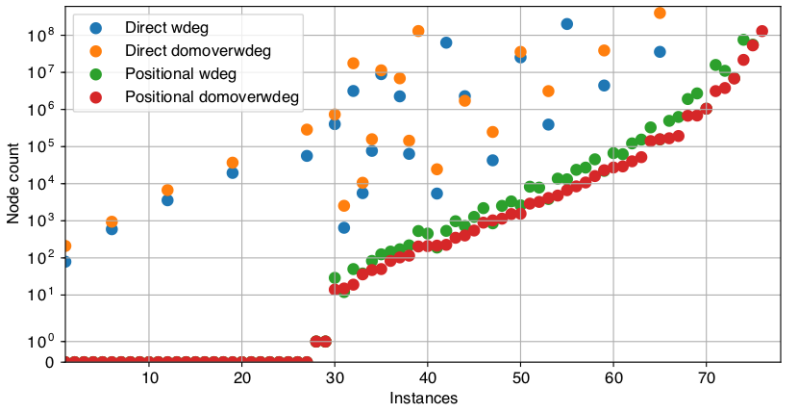

Fig. 5. Search nodes with weighted degree search heuristic with the Direct model vs the Positional model. Sorted by Positional-domoverwdeg. The timed-out instances are not plotted.

3.2 Channelled Approaches

At this point the Direct approach may seem hopeless. The Positional approach gives much better performance with the static variable ordering and with weighted degree heuristics in comparison.

In [17], after considering several options the best model is found to be a channelled model, with a smallest domain first variable ordering heuristic. In that paper the authors do not break the reflection symmetry in any of their

models, in this paper we would like to break this symmetry. Since each model comes with a symmetry breaking constraint and since these are incompatible, we experiment with each one separately. In Figure 6, these models are denoted by 'SDF sym:P' when the symmetry breaking constraint of the Positional model is used, and by 'SDF sym:D' when the Direct one is used. We compare the performance of the channelled models with the performance of the best model so far: Positional-domoverwdeg. The results are given in Figure 6.

Interestingly, the performance of the smallest domain first heuristic on the channelled model seems to depend heavily on which symmetry breaking constraint is used. The 'SDF sym:P' version gives better results than Positional-domoverwdeg, whereas 'SDF sym:D' is much worse overall.

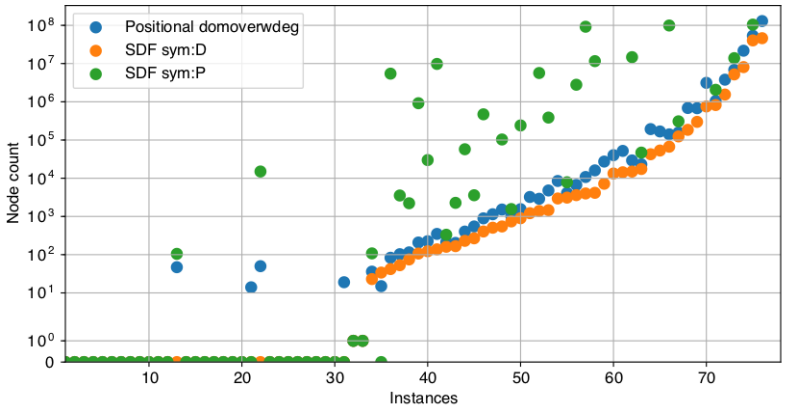

Fig. 6. Search nodes with the smallest domain first heuristic on channelled models vs Positional-domoverwdeg. Sorted by Positional-domoverwdeg. The timed-out instances are not plotted.

Figure 7 gives the difference between the 'branch:D sym:D cons:Both' and 'branch:P sym:P cons:Both' variations, where the first one uses a static variable ordering on the Direct model and the second one on the Positional model. Each model uses the corresponding symmetry breaking constraint and both sets of problem constraints. Hence, in comparison to the results given in Figure 4, these two variations use the same search order. However the results are inverted: branching on the Direct variables in the channelled model gives significantly better results than branching on the Positional variables. The Direct model by itself performs worse than the Positional model, yet branching on the Direct model performs significantly better than branching on the Positional model when the two models are channelled.

For reference, Table 1 provides detailed search node results across the best six models. It is evident that the static variable ordering on the direct variables

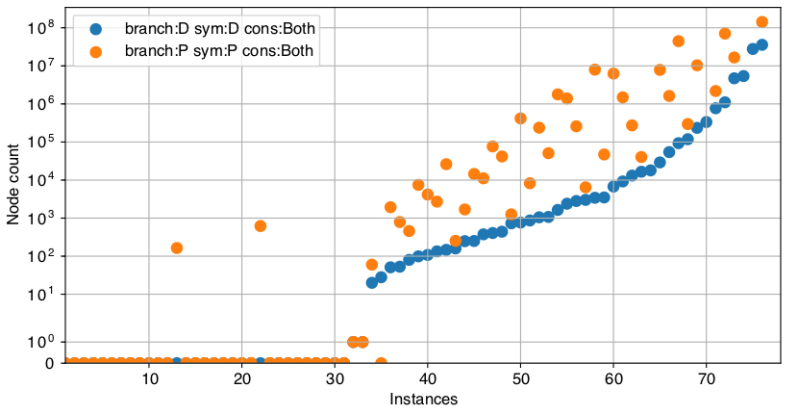

Fig. 7. Comparing the number of search nodes between static variable ordering on the Direct variables vs the Positional variables. Sorted by the 'branch:D sym:D' variation.

results in the smallest number of search nodes. Further reductions are obtained by using the symmetry breaking constraints of the direct model and keeping both sets of constraints.

3.3 Comparison against `BC_MINISAT_ALL`

We compare the performance of one of the best CP models to that of an AllSAT solver, BC_MINISAT_ALL [19]. We use Savile Row to produce a SAT encoding for the Direct model, the Positional model, and a channelled model with the symmetry breaking constraint from the Positional model. We only experiment with these three variations for SAT, since variable ordering heuristics cannot be specified like they can in a constraint solver.

The SAT encoding for the Direct model times out on eleven of 77 instances with a 4-hour timeout, and the Positional model times out on four. The channelled model performs better than either model in SAT as well, it solves all instances in 13,266 seconds in total. In comparison the channelled constraint model with the 'branch:D sym:P cons:P' variation (branching on Direct variables, symmetry breaking using the Positional constraint, and only using the Positional problem constraints) solves all instances in 7,408 seconds in total.

4 Conclusion

In this paper we have studied the performance of two different viewpoints of Langford's Problem, both separately and channelled together. Earlier work has demonstrated the utility of the channelled approach, but in contrast to previous findings, we found that the most effective search strategy to exploit the

Instance	Positional dom-wdeg	branch:D sym:D cons:Both	branch:D sym:D cons:P	branch:D sym:P cons:Both	branch:D sym:P cons:P	SDF sym:D
02_06	15	28	28	**0**	12	34
02_07	214	**160**	**160**	163	163	161
02_08	1,026	**730**	**730**	740	740	741
02_09	4,112	**3,015**	**3,015**	3,075	3,075	3,111
02_10	23,064	**16,526**	**16,526**	16,758	16,758	17,455
02_11	157,243	**118,165**	**118,165**	119,791	119,791	124,268
02_12	1,050,620	**768,853**	**768,853**	779,204	779,204	822,842
02_13	6,843,313	**4,705,524**	**4,705,524**	4,753,110	4,753,110	5,202,212
02_14	53,422,836	**35,680,214**	**35,680,214**	36,123,258	36,123,258	40,377,181
03_07	36	**20**	24	24	27	23
03_08	115	80	81	87	87	**75**
03_09	402	248	248	279	279	**230**
03_10	1,550	**867**	**867**	999	999	898
03_11	6,678	**3,495**	**3,495**	3,766	3,766	3,609
03_12	29,239	**13,127**	**13,127**	14,978	14,978	14,937
03_13	140,685	**54,281**	**54,281**	57,563	57,563	66,819
03_14	679,085	**235,009**	**235,009**	257,019	257,019	298,246
03_15	3,789,394	**1,102,150**	**1,102,150**	1,171,173	1,171,173	1,546,663
03_16	21,707,561	**5,384,480**	**5,384,480**	5,758,755	5,758,755	8,079,517
03_17	129,546,336	**27,773,357**	**27,773,357**	28,875,464	28,875,464	46,099,072
04_08	47	**0**	20	40	45	**0**
04_09	103	**53**	**53**	85	85	**53**
04_10	225	134	134	198	198	**123**
04_11	892	**375**	**375**	563	563	404
04_12	2,902	**1,075**	**1,075**	1,515	1,515	1,390
04_13	10,774	**2,816**	**2,816**	3,873	3,873	3,990
04_14	40,113	**9,186**	**9,186**	12,480	12,480	13,291
04_15	167,448	**29,091**	**29,091**	38,974	38,974	53,075
04_16	688,349	**93,089**	**93,089**	124,407	124,407	184,673
04_17	3,131,087	**334,480**	**334,480**	444,709	444,709	755,651
05_09	14	**0**	**0**	**0**	**0**	**0**
05_10	50	**0**	**0**	61	64	**0**
05_11	203	**108**	112	179	179	167
05_12	547	**251**	**251**	365	365	268
05_13	1,517	**440**	**440**	679	679	547
05_14	4,782	**1,051**	**1,051**	1,580	1,580	1,459
05_15	16,045	**2,391**	**2,391**	3,723	3,723	4,151
05_16	51,894	**6,743**	**6,743**	9,527	9,527	14,315
05_17	192,892	**17,974**	**17,974**	24,685	24,685	42,428
06_10	19	**0**	**0**	**0**	**0**	**0**
06_11	83	51	52	85	89	**42**
06_12	208	**99**	101	151	151	107
06_13	349	147	150	233	233	**138**
06_14	1,142	**406**	**406**	605	605	507
06_15	3,232	**765**	766	1,203	1,203	1,218
06_16	8,475	**1,642**	**1,642**	2,443	2,443	2,954
06_17	27,293	**3,424**	**3,424**	5,165	5,165	7,170
Mean	4,718,175	**1,624,811**	1,624,812	1,672,633	1,672,633	2,207,366
Sum	221,754,209	**76,366,120**	76,366,156	78,613,734	78,613,761	103,746,215

Table 1. Search nodes for our best six models of Langford's Problem.

channelled models is a static variable ordering on the viewpoint that, in isolation, is weaker. Our conjecture is that, through the channelling constraints, this static variable ordering produces a high quality dynamic variable ordering on the second viewpoint.

In future work, we will investigate whether this finding can be translated to other problem classes and whether our findings also hold with other constraint solvers.

Acknowledgements This work was supported by EPSRC EP/P015638/1. We thank our anonymous reviewers for helpful comments.

References

1. Akgün, Ö.: Extensible automated constraint modelling via refinement of abstract problem specifications. Ph.D. thesis, University of St Andrews (2014)
2. Akgün, Ö., Frisch, A.M., Gent, I.P., Hussain, B.S., Jefferson, C., Kotthoff, L., Miguel, I., Nightingale, P.: Automated symmetry breaking and model selection in Conjure. In: International Conference on Principles and Practice of Constraint Programming. pp. 107–116. Springer (2013)
3. Akgün, Ö., Gent, I.P., Jefferson, C., Miguel, I., Nightingale, P.: Breaking conditional symmetry in automated constraint modelling with Conjure. In: ECAI. pp. 3–8 (2014)
4. Akgün, Ö., Miguel, I., Jefferson, C., Frisch, A.M., Hnich, B.: Extensible automated constraint modelling. In: Proceedings of theTwenty-Fifth AAAI Conference on Artificial Intelligence. pp. 4–11. AAAI Press (2011)
5. Boussemart, F., Hemery, F., Lecoutre, C., Sais, L.: Boosting systematic search by weighting constraints. In: ECAI. vol. 16, p. 146 (2004)
6. Cheng, B., Choi, K.M.F., Lee, J.H.M., Wu, J.: Increasing constraint propagation by redundant modeling: an experience report. Constraints **4**(2), 167–192 (1999)
7. Frisch, A.M., Grum, M., Jefferson, C., Hernández, B.M., Miguel, I.: The Essence of Essence. Modelling and Reformulating Constraint Satisfaction Problems pp. 73–88 (2005)
8. Frisch, A.M., Grum, M., Jefferson, C., Hernández, B.M., Miguel, I.: The design of Essence: A constraint language for specifying combinatorial problems. In: IJCAI. pp. 80–87 (2007)
9. Frisch, A.M., Harvey, W., Jefferson, C., Martínez-Hernández, B., Miguel, I.: Essence: A constraint language for specifying combinatorial problems. Constraints **13**(3), 268–306 (2008)
10. Gent, I.P., Jefferson, C., Miguel, I.: Minion: A fast scalable constraint solver. In: ECAI. vol. 141, pp. 98–102 (2006)
11. Hnich, B., Smith, B.M., Walsh, T.: Dual modelling of permutation and injection problems. Journal of Artificial Intelligence Research **21**, 357–391 (2004)
12. Hnich, B., Walsh, T.: Why channel? multiple viewpoints for branching heuristics. Modelling and Reformulating Constraint Satisfaction Problems p. 94 (2003)
13. Nightingale, P., Akgün, Ö., Gent, I.P., Jefferson, C., Miguel, I.: Automatically improving constraint models in Savile Row through associative-commutative common subexpression elimination. In: CP. pp. 590–605. LNCS 8656, Springer (2014)

14. Nightingale, P., Akgün, O., Gent, I.P., Jefferson, C., Miguel, I., Spracklen, P.: Automatically improving constraint models in Savile Row. Artificial Intelligence **251**, 35–61 (2017). https://doi.org/10.1016/j.artint.2017.07.001
15. Nightingale, P., Spracklen, P., Miguel, I.: Automatically improving SAT encoding of constraint problems through common subexpression elimination in Savile Row. In: CP. pp. 330–340. LNCS 9255, Springer (2015)
16. Smith, B.M.: Comparing dual viewpoints in permutation problems. Constraint Modelling and Reformulation (ModRef'09) p. 147
17. Smith, B.M.: Modelling a permutation problem (2000)
18. Smith, B.M.: Dual models of permutation problems. In: International Conference on Principles and Practice of Constraint Programming. pp. 615–619. Springer (2001)
19. Toda, T., Soh, T.: Implementing efficient all solutions sat solvers. Journal of Experimental Algorithmics (JEA) **21**, 1–12 (2016)
20. Walsh, T.: Permutation problems and channelling constraints. In: International Conference on Logic for Programming Artificial Intelligence and Reasoning. pp. 377–391. Springer (2001)