Decision Making - ex 2

Filippo Brajucha, Youssef Hanna

October 2023

1 nQueens

1.1 Table

n	Alldifferent GC		Decomposition	
	Fails	**Time**	**Fails**	**Time**
28	78.847	1s 398ms	417.027	32s 508ms
29	31.294	639ms	212.257	17s 881ms
30	1.588.827	27s 79ms	7.472.978	11m 8s

1.2 Comment

Comment briefly on how the solver performance changes from one (decomposed or non-global) model to the global model, along with a justification.

Observing the data, it can be seen how the execution times and the number of failures change using a model based on global constraints rather than one based on their decomposition.
Looking at this specific problem we can observe how the time values increase by a multiplier greater than 20x, while failures increase by approximately 5x.
These differences are attributed to the propagation efficiency (better propagation algorithm) and propagation speed (operational advantage) that distinguishes global constraints. They allow for a much quicker and more efficient reduction of the search space compared to a regular constraint thanks to these qualities.
This operation is facilitated by the fact that global constraints manage to narrow the gap between the problem and the model, making the problem more understandable to the solver machine.

2 Poster Placement

2.1 Table

Instance	**Naive Model**		**Global Model**	
	Fails	**Time**	**Fails**	**Time**
19x19	1.678.013	16s 406ms	300.649	2s 868ms
20x20	2.504.120	25s 81ms	2.030	291ms

2.2 Comment

Comment briefly on how the solver performance changes from one (decomposed or non-global) model to the global model, along with a justification.

The results obtained using one model compared to another differ significantly. Between the naive model and the global model, there are significant differences. With both input datasets, the search times drop significantly. In the first case, they become about 1/8, while in the second case, they decrease by 98%.
The number of failures also decreases rapidly, moving from the order of millions to orders of magnitude smaller. In the case of the 20x20 grid, it goes from 2.5 million to 2 thousand.
These data highlight how much more powerful and efficient the use of global constraints is. They allow for a much faster and more effective reduction of the search space compared to the naive model or, generally, when using regular constraints.
This improvement in terms of efficiency is certainly due to the embedded specialized propagation, which exploits the substructure with a much more efficient solver machine, both in terms of strong inference in propagation and propagation speed.

3 Sequence Puzzle

3.1 Table

n	**Base**		**Base + Implied**		**Global**		**Global + Implied**	
	Fails	**Time**	**Fails**	**Time**	**Fails**	**Time**	**Fails**	**Time**
500	617	18s 711ms	495	12s 321ms	989	449ms	493	515ms
1000	1.247	1m 57s	995	54s 793ms	1.989	1s 415ms	993	2s 446ms

3.2 Questions

1. Going from Base → Global, and going from Base + Implied → Global + Implied: what is the main advantage of using a global constraint? Why?
2. Is there an implied constraint that now becomes redundant in the Global + Implied model? Why?

3.3 Answers

1. The main advantage of using a global constraint model is the execution speed. Whether transitioning from ”base” to ”global” or from ”base+implied” to ”global+implied”, a significant decrease in execution time can be observed.
   What distinguishes global constraints is that they have a much more effective propagation and resolution algorithm in terms of failures and are much more aligned with the model because they understand better how the problem is structured.
   In the case of sequence puzzle problem, we can see how, despite sometimes being more complex to use (*as described in chapter 11.7.2 of the handbook*), it is beneficial to use them, especially when considering the reduction in resolution time, which drops rapidly from several tens of seconds (and even minutes) to just a few hundred milliseconds.
   However, in the transition from ”Base + Implied” to ”Global + Implied”, a decrease in propagation efficiency is notable, as failures remain nearly the same. This is likely due to the implied constraints that are redundant.
   In terms of time, the ”Base” model is more expensive than ”Base + Implied” because the latter includes implied constraints that significantly reduce the search space more rapidly.
   Generally, using a model with global constraints is also advantageous in terms of computational cost since using traditional constraints involves introducing a ”forall” loop with 2 variables (i and j, each of these used by a sum) ranging from 0 to n-1, which certainly increases the execution time.
   For this reason, it can be said that the ”Base” model, even though it uses implied constraints, will still be slower.

2. We have understood through reading the documentation that the "global cardinality" constraint comprehends the first implied constraint.
   The solver algorithm for the "global cardinality" takes two inputs:

   (a) The list of variables (`VARIABLES`)
   (b) The variables for which to keep track of the number of occurrences (`VALUES`)

   Then it applies constraints, including the following:

   ```
   VALUES.noccurrence <= |VARIABLES|
   ```

   which checks that the number of occurrences (of VALUES) is less than or equal to —VARIABLES— (the length of the VARIABLES list).
   The purpose of this constraint is to ensure that each value, VALUE[i].value (with i ranging from 1 to the length of VALUES), must occur exactly VALUES[i].noccurrence times at position i in VARIABLES.
   In this way, one can observe how the summation of occurrences in VARIABLES will be exactly equal to n (the size of VARIABLES), and thus, it can be demonstrated why the first implied constraint is redundant.

   So we can be sure that the implied constraint

   ```
   sum(j in 0..n-1)(seq[j]) == n
   ```

   is redundant and it increases the search time because the solver makes the same operation twice.