Decision Making with Constraint Programming

Exercise 1

Dessì Leonardo [0001141270]

Spini Riccardo [0001084256]

Corso di laurea magistrale in Informatica

Alma Mater Studiorum
Università di Bologna

October, 2023

1 | N-Queens problem

The n-queens problem is about finding how many different ways queens can be placed on a chessboard so that none attack each other.
The table 1.1 shows the number of solutions and failures of the six models implemented. Analysing the results we can see models return more solutions and failures by increasing n.

n	#sols	r	rc1	rc2	rc3	alldiff
8	92	891	500	593	864	254
9	352	4 262	2 656	2 771	4 458	849
10	724	23 291	14 003	13 585	23 201	3 722
12	14 200	773 550	353 151	380 556	820 087	75 823

Tabella 1.1: Result with simmetry

n	#sols	alldiffsym
8	12	89
9	46	272
10	92	944
12	1 787	16 240

Tabella 1.2: Results with symmetry breaking

1.1 | What is happening when going r → rc1 → alldiff ? Why?

- In the case of the model denoted by "r", the constraints are that:

  1. no queen attacks another queen on the same column

     ```
     constraint alldifferent(rows)::domain_propagation;
     ```

  2. no queen attacks another queen on its diagonal

     ```
     constraint forall(i,j in 1..n where i<j)(abs(rows[i]-rows[j]) != abs(i-j));
     ```

  These two constraints are too general.

- The combined RC model is a model that mixes the model in which the decision variables are $\forall i, X_i \in [1..n]$ and the model in which the decision variables are $\forall i, j.\ B_{ij} \in [0..1]$ and binds the two models using a Channeling Constraint.

  The RC model performs better than the Row model because we combine the Row model with another model with more constraints. This implementation allows the

new RC constraints to exploit the row constraints and vice versa, increasing propagation.

- The "*Alldifferent*" model is a model derived by changing the diagonal attack constraint in the "*r*" model with two global `alldifferent` constraints. Because global constraints are implemented with ad-hoc algorithms and not with simple propagation as in normal constraints, these are more efficient and the search space decreases. This can also be observed from the results obtained.

1.2 | What is happening when going rc1 → rc2 → rc3 ? Why?

- The RC combined model can be implemented in several ways, in fact, it can be seen that in the RC1 Model, there are redundant constraints that if removed leave the solutions unchanged. Removing the redundant constraints, however, alters the performance of the solver, as can be seen in the table 1.1.
- The RC2 Model is derived by eliminating two `alldifferent` constraints from RC1 Model. This is possible thanks to the presence of the channeling constraint, which defines that the values of X_i and Y_j must differ. However, when the two global constraints are eliminated, the solver's performance deteriorates. This is because global constraints are implemented with specialized algorithms that outperform solutions lacking such global constraints. In essence, by removing these two constraints, the search space expands, leading to a decrease in solver effectiveness.
- Similarly, the RC3 Model is derived by removing an implied constraint that gives additional information about diagonal attacks. Once more, when we eliminate even this implied constraint, we notice a more evident drop in the solver's performance, resulting in an increased rate of failure.
- Consequently, these results highlight the significance of the modeling phase. It becomes evident that a modeler's task extends beyond mere functionality; it is crucial to craft a model that is not only effective but also efficient, employing global constraints and implied constraints to optimize its performance.

1.3 | What is happening when going alldiff → alldiffsym? Why?

As we did between model "*r*" and "*RC*", we combine the "*Alldifferent*" model with "*Symmetry breaking*" by removing all symmetrically equal solutions from *Alldifferent* and adding the constraint: `symmetry_breaking_constraint`. Symmetry breaking constraints impose

an order or relationship between the original solution and all the solutions obtained by applying various permutations. By doing so, they effectively eliminate symmetrical or equivalent solutions from consideration, leaving only one representative solution from each symmetrical group.
In particular constraints such as `lex_lesseq`, introduce restrictions to ensure that there are no queens threatening the horizontal, vertical, or diagonals.
We are therefore reducing the search space, which is why failures are decreasing.

2 | Sequence Problem

In this problem, we need to solve a puzzle where we have to find a sequence of n integers $X_0, \ldots, X_{n-1}$ that contains values between 0 and $n-1$, in a way that any value i appears X_i times in the sequence. Table 2.1 reports failures and total time of two models implemented for the Puzzle problem.

n	Base		Base + Implied	
	fails	time	fails	time
500	618	34s	495	25s
1000	1 743	1h 21m	995	1m 38s

Tabella 2.1: Sequence results

2.1 | What is happening when going base → base+implied? Why?

The second model adds two implicit constraints $\sum_i X_i = n$, $\sum_i X_i * i = n$ to the basic model. The implicit constraints make it possible to reduce the branches of the decision tree and thus reduce the search space and increase the performance of the model.
When the solver identifies implied constraints, it can eliminate certain variable assignments or combinations of values that are inconsistent with the problem's constraints, reducing the number of possibilities to explore.
Implied constraints can be discovered through various techniques, such as constraint propagation, or domain reduction. This, in turn, can significantly improve the efficiency and speed of the constraint satisfaction process.