Scheduling Match Races for Sailing

Craig Macdonald, Ciaran McCreesh, Alice Miller and Patrick Prosser
University of Glasgow, Scotland ; c.mccreesh.1@research.gla.ac.uk

University of Glasgow

In sailing match races, **skippers race in pairs**. Each skipper races against each other skipper once, in a **round-robin** tournament. The order of matches in rounds is important: for example, a skipper who sails last in one round must not sail first in the following round. When there are fewer boats than skippers, **boats must be shared** and we would like to **minimise the number of boat changes**. We used **constraint programming** to develop **fairer schedules** for these competitions.

Sailing Match Races

In a match race, skippers races against each other in pairs. There are **several hundred events** per year, each involving up to ten of the 1,500 internationally registered skippers.

Round-Robin Schedules

Sailing match race schedules are provided as **downloadable templates**, using numbers instead of skipper names. In each round, pairs of skippers race in a sequence of matches.

Round	Matches			Round	Matches		
1	(1,2)	(3,4)	(5,6)	5	(4,6)	(2,7)	(3,5)
2	(1,3)	(5,7)	(2,6)	6	(4,7)	(1,5)	(3,6)
3	(3,7)	(1,6)	(2,4)	7	(4,5)	(2,3)	(1,7)
4	(6,7)	(1,4)	(2,5)				

The International Sailing Federation specifies what makes a legal match race schedule. There are 13 criteria, and all existing schedules were **produced by hand**. This is a daunting task: **most are illegal**, many are missing, and some that are legal are **not as fair for competitors** as they could be.

Constraint Programming

Constraint programming lets us describe a **model**, in terms of **variables**, a set of **constraints** between the variables, and an **objective**. We do *not* need to provide an algorithm: we simply gave our model to the **Choco Solver** to solve.

For match racing, this let us focus on **understanding the problem** rather than designing an algorithm, and allowed us to develop a model **incrementally**, by adding in constraints in stages. Being able to use **expressive constraints** and **natural variable types** made our model easier to develop, easier to understand, and easier to verify.

Channelling Between Perspectives

When scheduling match races, some rules are temporal, some are described from a skipper's perspective, and some relate to rounds.

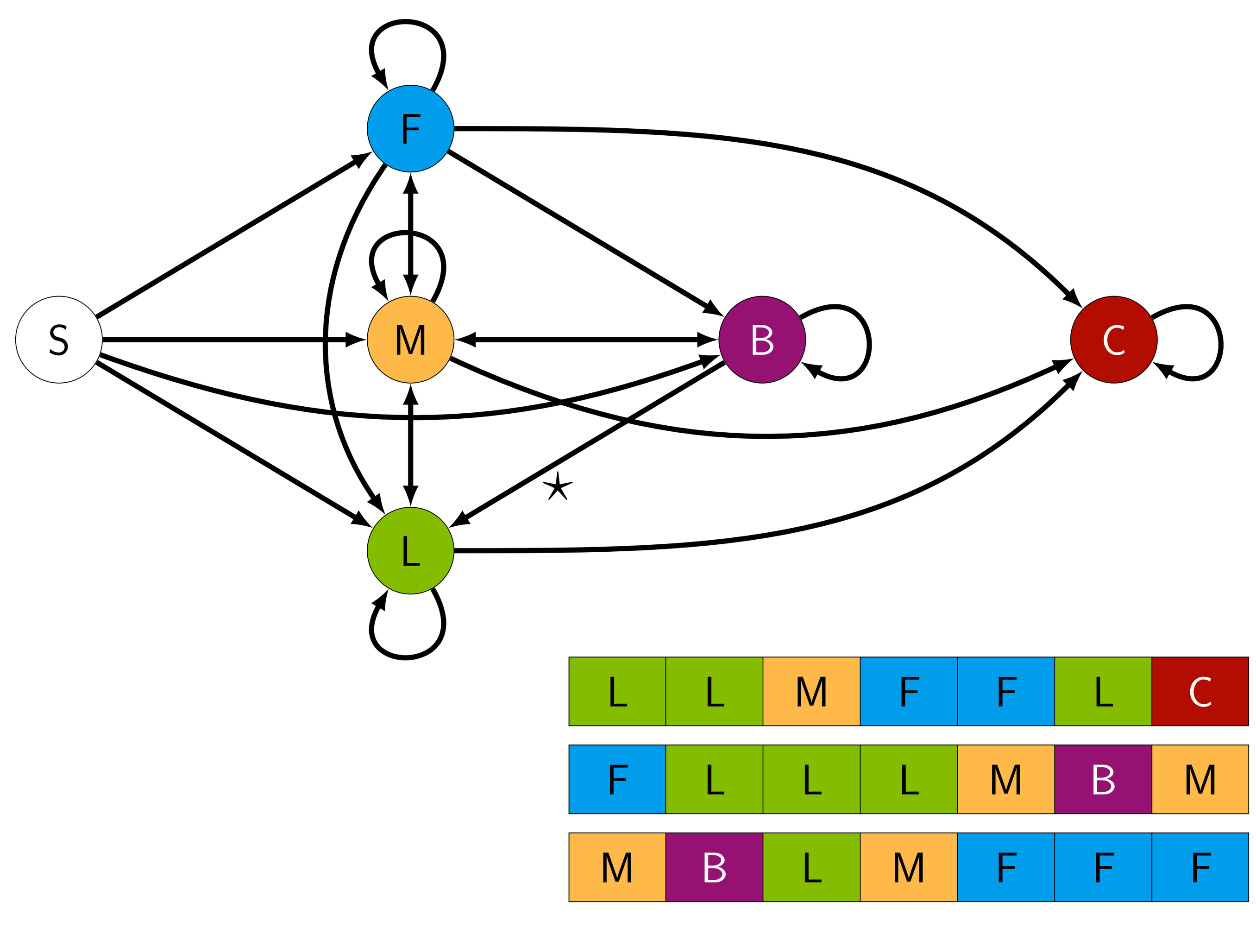
Constraint programming allows us to have **more than one model** of the problem, and we can express each constraint on whichever model is **most natural**. We then link the models using **channelling** constraints.

We used four models simultaneously for match racing. This was both **simpler** than a single model, and gave **better performance**.

Modelling Patterns in Sequences

If a skipper is last out in one round, they must not go out first in the next round, to give them time to recover. Similar rules govern byes and boat changes. We model this using a sequence of states for each skipper, specifying whether that skipper is in first position, last, in the middle, in a bye, or has completed all their matches, for each round. The diagram shows which transitions are valid, with three examples—the automaton is used directly, as a *regular* constraint on the sequence.

This kind of constraint is also used for scheduling suitable patterns of **day and night shifts** for staff rotas, and specifying **permitted sequences of jobs** on a production line.



Craig Macdonald, Ciaran McCreesh, Alice Miller, Patrick Prosser: **Constructing Sailing Match Race Schedules: Round-Robin Pairing Lists.** CP 2015: 671-686.