Round robin scheduling with CP-SAT from Google OR tools

Asked 1 year, 10 months ago Modified 1 year, 9 months ago Viewed 236 times

0

I am creating round robin tournament fixtures with CP-SAT from google OR tools in python.

Problem: There are multiple teams about 150 split in to different groups and divisions in a league. The teams share ground and ground also has constraint about some days it will not be available. I need to make a fairly allocated round robin groups where teams play home and away leg matches. Teams do not want to plan more than 1 or 2 home matches consecutively. When possible team wants to finish playing against all oppositions (with combination of home/away) before playing each other again.

Solution:

Currently I using, boolean variables g_h_o_d (ground_home_opposition_date). And add constraints such as:

- No two teams plays on the same ground on same date
- All home teams play exactly one match against all oppositions
- Team play atmost one match on a given date.

Result: This works well, but the result is not distributed well enough as in I could have two team (x and y) playing against each in two consecutive days or teams playing all home matches before starting their away. Any clue, what constraints I can add to prevent this?

```
from ortools.sat.python import cp_model
import logging
import sys
import math

def match_date(n):
    return n[-1]

class SolutionPrinter(cp_model.CpSolverSolutionCallback):
  """Print intermediate solutions."""

  def __init__(self, matches):
      cp_model.CpSolverSolutionCallback.__init__(self)
      self._matches = matches

  def on_solution_callback(self):
      result = []
      match_count = 0
      for match in self._matches:
        h,o,d = match[0],match[1],match[2]
        if self.Value(self._matches[(h,o,d)]):
          match_count += 1
          result.append(match)
      logging.info('Match Count: %i' % match_count)
      for match in sorted(result, key=match_date):
        print (f"{match[2]}: {match[0]} X {match[1]}")
      self.StopSearch()
```

```
def teams_on_a_day_constraint(model, matches):
  # team plays atmost one match in a day
  # regardless of ground or opposition
  constraints = {}
  for match in matches:
    h,o,d = match[0],match[1],match[2]
    try:
      constraints[f"{h}_{d}"].append(matches[h,o,d])
    except KeyError:
```

or-tools cp-sat

Share Improve this question Follow

edited Dec 20, 2023 at 6:24
Laurent Perron
11.2k 1 10 27

asked Dec 10, 2023 at 12:56
wantro
355 1 7 19

1 Answer

Sorted by: Highest score (default)

2

In the sports_scheduling example, we force the scheduling to be separated in 2 half seasons.

You could also add rolling constraints for each team saying that the sum over X fixtures of the 2 booleans corresponding to the same opponent must be <= 1.

Share Improve this answer Follow

answered Dec 10, 2023 at 14:18
Laurent Perron
11.2k 1 10 27

Sign up to request clarification or add additional context in comments.

1 Comment

Add a comment

wantro Over a year ago

Thanks Laurent, thanks for the link, I am keen to understand the `// Forbid sequence of 3 homes or 3 aways.` I will look into this. I did the rolling constraint but was thinking if there is a better way to do it. Also, because of the constraints, these two half, should be more like a "soft limit", as in maximise the gap between the sshedules. That's why I was thinking if there is any way I can use an Integer variable somehow.

0 Reply

Start asking to get answers

Find the answer to your question by asking.

Ask question

Explore related questions

or-tools cp-sat

See similar questions with these tags.