COMP6203 Intelligent Agents
Lab Example Code

Jan Buermann
j.buermann@soton.ac.uk

2024

Contents

Lab #2

2.1 Exercise 1: Creating a Schedule

2.1.1 Exercise 1.1: A Very Simple Scheduling

The simple scheduling just iterates through all trades and for every trade through all vessels and sees if the trade can be appended to the current schedule without violating the constraints. Once a ship has been found that can transport the trade it will simply propose that.

```python
class MyCompany(TradingCompany):

    def propose_schedules(self, trades):
        schedules = {}
        scheduled_trades = []
        i = 0
        while i < len(trades):
            current_trade = trades[i]
            is_assigned = False
            j = 0
            while j < len(self._fleet) and not is_assigned:
                current_vessel = self._fleet[j]
                current_vessel_schedule = schedules.get(current_vessel,
                ↪   current_vessel.schedule)
                new_schedule = current_vessel_schedule.copy()
                new_schedule.add_transportation(current_trade)
                if new_schedule.verify_schedule():
                    schedules[current_vessel] = new_schedule
                    scheduled_trades.append(current_trade)
                    is_assigned = True
                j += 1
            i += 1
        return ScheduleProposal(schedules, scheduled_trades, {})
```

2.1.2 Exercise 1.2: Slightly Better Scheduling

The slightly better scheduling works similar to the simple scheduling but for every trade-vessel combination will try to find the best time to pick up and drop off the current trade within the scheduled order of pick-ups and drop-offs that are already scheduled for the vessel. the best time is determined here by using the shortest completion time (`completion_time`).

```
class MyCompany(TradingCompany):

    def propose_schedules(self, trades):
        schedules = {}
        scheduled_trades = []
        i = 0
        while i < len(trades):
            current_trade = trades[i]
            is_assigned = False
            j = 0
            while j < len(self._fleet) and not is_assigned:
                current_vessel = self._fleet[j]
                current_vessel_schedule = schedules.get(current_vessel,
                ↪  current_vessel.schedule)
                new_schedule = current_vessel_schedule.copy()
                insertion_points = new_schedule.get_insertion_points()
                shortest_schedule = None
                for k in range(len(insertion_points)):
                    idx_pick_up = insertion_points[k]
                    insertion_point_after_idx_k = insertion_points[k:]
                    for m in range(len(insertion_point_after_idx_k)):
                        idx_drop_off = insertion_point_after_idx_k[m]
                        new_schedule_test = new_schedule.copy()
                        new_schedule_test.⌟
                        ↪  add_transportation(current_trade,
                        ↪  idx_pick_up, idx_drop_off)
                        if (shortest_schedule is None
                                or new_schedule_test.completion_time()
                                ↪  < shortest_schedule.⌟
                                ↪  completion_time()):
                            if new_schedule_test.verify_schedule():
                                shortest_schedule = new_schedule_test
                if shortest_schedule is not None:
                    schedules[current_vessel] = shortest_schedule
                    scheduled_trades.append(current_trade)
                    is_assigned = True
                j += 1
            i += 1
        return ScheduleProposal(schedules, scheduled_trades, {})
```