Introducing a Foundational Framework for Intelligent Baseball Scheduling

Peiyuan Xu[1][2], Kyoungjin Lim[2], Antwan D. Clark[1][3], and Anton Dahbura[2][3]
[1] Applied Mathematics and Statistics Department, [2] Department of Computer Science, [3] Institute for Assured Autonomy
Johns Hopkins University

Abstract

Developing a fair and balanced baseball schedule is a challenging problem. This is because the appropriate balance between required and desired needs to be obtained while ensuring that these constraints are both applied and satisfied. Additionally, alternative measures need to be made to account for cases when certain combination of constraints applied together create an unsatisfactory schedule. This work addresses these limitations via first presenting a foundational framework for intelligent baseball scheduling that include the following.

1. Developed a Python-based constraint library that describes required constraints.
2. Modeled the incorporation of required constraints as a Markov decision process.
3. Developed a custom AI Gym environment that demonstrates this process, where we also leverage state-of-the art operations research (OR) tools (e.g., Google OR tools).

Our results using specific constraints for the High-A Central Baseball League demonstrate promise in this framework, which is built via the Open Artificial Intelligence (AI) Gym Environment.

Introduction

Scheduling for baseball leagues need to satisfy a variety of constraints that are organized by the following categories.

1. *Required (or immovable) constraints* – Constraints that must be satisfied to produce workable timetables.
2. *Desired (or movable) constraints* – Constraints that don't need to be satisfied to produce workable timetables.

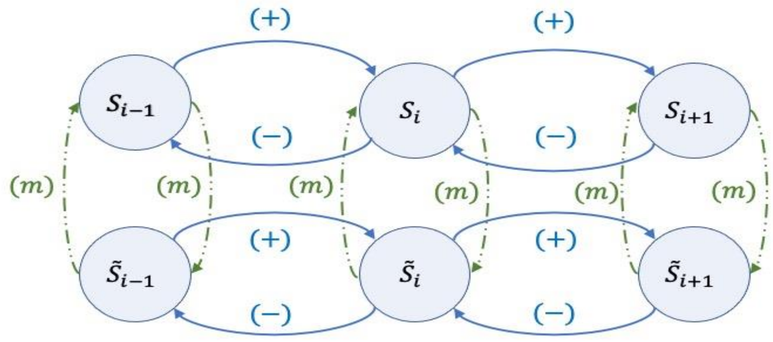
Each of these categories also contain other types of constraints including those based on the following: 1.) the uniqueness of home play; 2.) the number of games played within a specific region; 3.) how opponents are scheduled; and 4.) schedule fairness.

Previous advancements to this problem have been made via various combinatorial optimization methods such as semi-definite programming, integer programming, and heuristic methods. However, these demands do not completely consider real-time scheduling demands and changes that are typically league-specific.

Theoretical Development

We created the following Markov decision process (MDP) for immovable and movable constraints.

- The state space $\boldsymbol{S}$ includes the number *and type* of constraints added and satisfied to the baseball schedule.
- The action space $\boldsymbol{A}$ includes the number of constraints either added $(+)$, deleted $(-)$, or modified/substituted (m). If the constraints are immovable, the action is only substituted. Otherwise, the action can either be modified and/or substituted.
- The rewards $\boldsymbol{R}$ are defined as follows: 1.) +100 for each constraint added and satisfied and 2.) +50 for each constraint added and satisfied where at least one constraint is modified.



Transition Sketch of the Proposed MDP

AI Gym Framework

Initialization

1. Defines the initial number of immovable constraints and how many of these constraints are satisfied.
2. The Python constraints library is incorporated into this environment via providing each constraint with numerical unique identifiers (UIDs) to construct each state in the MDP.
3. The rewards are also defined in this step initializing the MDP.

Reset

The AI Gym environment is reset for further use.

Step

1. The actions add (+), remove (−), and modify ( ) are defined in this step where each constraint *and type of constraint* is added to the optimization problem.
2. Constraint satisfaction measures are enforced to ensure that each constraint and the collection of constraints are satisfied.
3. Incremental rewards are appropriated with each step in the environment as it goes from one state to the next.

Render

1. The states associated with the number and type of constraints added and satisfied are presented as output, which is associated with each state.
2. The accumulated rewards are also displayed along with the metrics computing the percentage of these constraints are satisfied.
3. The schedule is finalized and written to a timetable in comma separated value (CSV) format.

Testing Strategy

The goal of our testing strategy is to robustly test the proposed MDP process consisting of the following steps.

1. We incrementally step through the environment to determine if each state in the MDP is implemented correctly.
2. We recorded each state, action, and next state of the MDP where we checked to see if the constraints were added and satisfied successfully.

We do this for random types of immovable constraints (from the High A Central Baseball League requirements). Here, we also leveraged Google OR tools to perform optimization via a Python-based constraint library that we developed.

Results

The following is an example schedule (in Excel), where five immovable constraints were satisfied and added as a result of the MDP process.

1 to 10 of 756 entries Filter

serie	date	away_team	home_team
1	4/8/2022	Dayton Dragons	Beloit Sky Carp
1	4/8/2022	Fort Wayne TinCaps	Wisconsin Timber Rattlers
1	4/8/2022	Great Lakes Loons	South Bend Cubs
1	4/8/2022	Lake County Captains	Peoria Cheifs
1	4/8/2022	Lansing Lugnuts	Cedar Rapids Kernels
1	4/8/2022	West Michigan Whitecaps	Quad Cities River Bandits
1	4/9/2022	Dayton Dragons	Beloit Sky Carp
1	4/9/2022	Fort Wayne TinCaps	Wisconsin Timber Rattlers
1	4/9/2022	Great Lakes Loons	South Bend Cubs
1	4/9/2022	Lake County Captains	Peoria Cheifs

Show 10 per page 1 2 10 70 76

Conclusion

We demonstrated a foundational framework for intelligent baseball scheduling, where we incorporated various immovable constraints to illustrate the fidelity of the MDP. Some future explorations include more robust testing to include both required and desired constraints as well as exploring reinforcement learning strategies from this schema.

Additional Questions?

Peiyuan Xu (pxu11@jhu.edu), Kyoungjin Lim (klim30@jhu.edu)
Dr. Antwan D. Clark (aclark66@jhu.edu)
Dr. Anton Dahbura (anton.dahbura@jhu.edu)