The Travelling Tournament Problem

Comparing Exact and Heuristic Approaches

Group 5

Antal-Cionta, Ioan-Sorin

Chidambararajan Vijayalakshmi, Sibi Karthik

Fushada, Jessica

Verma, Gourisha

Vesalapu, Likhit

October 16, 2025

Table of Contents

1 Literature Study

The Travelling Tournament Problem (TTP) has attracted significant research interest because it models realistic constraints found in professional sports scheduling. The problem is known to be computationally challenging due to the combinatorial explosion in scheduling possibilities combined with constraints on home/away sequences and minimizing travel distances.

Exact approaches, such as integer linear programming (ILP) formulations, have been widely applied to solve small or medium-sized problem instances optimally. These methods rely on carefully crafted mathematical models to encode constraints and objective functions. Despite advances in solver technology and modeling techniques, ILP approaches struggle with scalability as the number of teams increases, often requiring prohibitive computation times.

To complement exact methods, a range of heuristic and metaheuristic algorithms have been developed. These approaches include simulated annealing, tabu search, genetic algorithms, and ant colony optimization. Heuristics trade off guaranteed optimality for practical run-time efficiency and scalability. They generate feasible solutions quickly and opportunistically explore neighborhoods of solutions to improve schedule quality. Simulated annealing, in particular, is notable for its ability to escape local minima via controlled randomization and temperature schedules, being effective in generating near-optimal feasible tournament schedules. In the case of genetic algorithms, a compact chromosome representation can be used, where each chromosome contains a permutation of teams and each gene is associated to a team. The Polygon Method is then applied to the chromosome, which generates a feasible schedule and significantly reduces the search space.

More recent research has focused on hybrid methods that combine ILP lower bounds with powerful heuristics to guide the search for promising solutions. These hybrids leverage structural properties of the TTP and problem decompositions, including the use of vehicle routing problem relaxations to improve bounds. Advances in computational power and algorithms have also triggered exploration of machine learning and reinforcement learning techniques for adaptive heuristic control in tournament scheduling.

Overall, the literature underscores the difficulty of TTP and the complementary roles of exact and heuristic methods. Our work builds on these foundations by implementing both Constraint programming and genetic algorithms approaches, testing them on clas-

sical benchmark instances like NL4 and comparing their effectiveness in solution quality and computational efficiency.

Key References:

1. Easton, K., Nemhauser, G., Trick, M. *The Traveling Tournament Problem—Description and Benchmarks.*

2. Pace, G., Raidl, G. *Randomized Construction Approaches to the Traveling Tournament Problem using Lower Bound Based Heuristics.*

3. Staats, P. *Generic Scheduling of Sports Tournaments.*

4. Siemann, M. R. *A Polyhedral Study of the Travelling Tournament Problem.*

5. Zhao, J., Xiao, M. *Practical Algorithms with Guaranteed Approximation Ratio for the Traveling Tournament Problem with Maximum Tour Length Two.*

6. Anagnostopoulos, A., et al. *A Simulated Annealing Approach to the Traveling Tournament Problem.*

7. Biajoli, F. L., Lorena, L. A. N. *Mirrored Traveling Tournament Problem: An Evolutionary Approach.*

8. Dimitsas, Angelos, et al. *A Pragmatic Approach for Solving the Sports Scheduling Problem*

2 Problem Formulation

The Travelling Tournament Problem considers N teams competing in a double round-robin tournament with $2(N-1)$ rounds. Each team plays every other team twice, once at home and once away. Teams play one game each round. The problem imposes constraints limiting consecutive sequences of home or away games to a minimum of L and maximum of U. Additionally, a no-repeat constraint prohibits teams from playing the same opponent in consecutive rounds. The tournament is modeled on a weighted complete graph where edges represent travel distances between teams. The objective is to minimize the sum of all team travels while respecting the constraints.

We came up with a set of constraints that differ from those from the researched literature:

- Enforcing that a team should move from the previous place it was last round. Also

by defining the position from round 0, we can tell where a team's home is:

$$y_{ij(k-1)} + y_{jsk} = 2 \qquad \forall \ \ i, j \in N$$

with:

$$y_{i,i,0} = 1 \qquad \forall \ \ i \in N,$$

$$y_{i,j,0} = 0 \qquad \forall \ \ i, j \in N, i \neq j$$

- Stating that only two teams should travel to the same location in a round:

$$y_{i,j,k} + y_{l,j,k} = 2 \qquad \forall \ \ i, j, l \in N$$

- The number of games per round should be defined as the following:

$$\frac{\sum_i^N \sum_j^N y_{i,j,k}}{N} = \frac{1}{2} \qquad \forall \ \ k \in \{1, ..., N\}$$

We ultimately decided to base our problem formulation on the papers we referred to in the previous section, as the constraints we came up with are not enough and we struggled to model more. There are a few benefits of having an exact IP formulation for TTP: solvers are capable of producing optimal solutions. Additionally, problem formulation with IP allows us to explicitly model all constraints and precisely calculate the distance cost with binary variables. However, the nature of IP doesn't scale very well: the bigger the instance is, the larger the search space and the more difficult to solve optimally. As we will show in our results section, the time to solve grows exponentially as the number of teams increases and in some cases, the algorithms timed out. The exact methods are not practical for large cases as we often see in real life.

In contrast, heuristic approaches such as genetic algorithms, tabu search and simulated annealing among others are built to address the scalability problem; they aim to quickly generate good solutions within short time frames, even for large instances. As a drawback, they do not guarantee optimality like IP does, but for practical purposes in real-world applications, the time factor may often be more important than necessarily finding the best solutions.

2.1 Parameters

- N: number of teams
- $D = [d_{ij}]$: distance / cost between teams, where

$$d_{ii} = 0 \quad \forall i, \quad d_{ij} = d_{ji} \quad \forall i, j.$$

- L: minimum length of consecutive home or away games
- U: maximum length of consecutive home or away games
- $k \in \{1, \ldots, 2(N-1)\}$: rounds / game days.
- $i, j, s, t \in \{1, \ldots, N\}$: team indexes.

2.2 Decision Variables

- $x_{k,i,j} \in \{0, 1\}$ for all k, i, j with $i \neq j$:
  1 if team i plays against team j at home in round k, 0 otherwise.
- $y_{i,s,t} \in \{0, 1\}$ for all i, s, t with $s \neq t$:
  1 if team i travels from location s to location t, 0 otherwise.

2.3 Objective

$$\min \quad \sum_i \sum_s \sum_{t \neq s} d_{s,t} y_{i,s,t}$$

The objective minimizes the total travel distance of all teams throughout the tournament while taking into account the constraints. For each team i, the model adds the distances from location s to t when the team travels between them. $d_{s,t}$ refers to the distance between locations, and the decision variable $y_{i,s,t}$ determines whether team i traveled.

2.4 Constraints

$\sum_{j \neq i}(x_{k,i,j} + x_{k,j,i}) = 1 \forall\, k, i$

Each team plays exactly one game in every round.

$\sum_k x_{k,i,j} = 1 \forall\, i,\ j \neq i$

Each team must play exactly once at home and once away against each other team.

$$\mathrm{x}_{k,i,j} + x_{k+1,j,i} \leq 1 k = 1, \ldots, 2N-3, \ \forall\, i \neq j$$

No teams can play against each other in consecutive rounds.

$$\sum_{l=0}^{U} \sum_{j \neq i} x_{k+l,i,j} \leq U k = 1, \ldots, 2(N-1) - U, \ \forall\, i$$

A team may play at most U consecutive home games.

$$\sum_{l=0}^{U} \sum_{i \neq j} x_{k+l,i,j} \leq U k = 1, \ldots, 2(N-1) - U, \ \forall\, j$$

A team may play at most U consecutive away games.

$$\mathrm{y}_{i,s,t} \geq x_{k,s,i} + x_{k+1,t,i} - 1 k = 1, \ldots, 2N-3, \ s \neq i, \ t \neq \{s, i\}$$

Handles travel between two consecutive away games.

$$\mathrm{y}_{i,i,t} \geq \sum_{j \neq i} x_{k,i,j} + x_{k+1,t,i} - 1 k = 1, \ldots, 2N-3, \ t \neq i$$

Handles travel from a home game to an away game.

$$\mathrm{y}_{i,t,i} \geq x_{k,t,i} + \sum_{j \neq i} x_{k+1,i,j} - 1 k = 1, \ldots, 2N-3, \ t \neq i$$

Handles travel from an away game to a home game.

$$\mathrm{y}_{i,i,t} \geq x_{1,t,i} \forall\, i, \ t \neq i$$

Handles travel to the away location for the first round if the game is away.

$$\mathrm{y}_{i,t,i} \geq x_{2(N-1),t,i} \forall\, i, \ t \neq i$$

Handles travel back to home for the last round if the game is away.

3 Solution Methods

3.1 Overview

We implemented two complementary solution approaches:

- **Exact:** Constraint programming using Google's OR-Tools CP-SAT solver, a hybrid solver that combines constraint propagation with SAT (Boolean satisfiability) solving and integer programming techniques.
- **Heuristic:** For the heuristic approach, we used a Genetic Algorithm to find good solutions quickly. It evolves a population of valid schedules through selection, crossover, and mutation, minimizing total travel distance.

Project Code Repository: https://github.com/likhit2002/Optimization$_T$TP.git

3.2 Exact Solution Approach: CP-SAT

Although heuristics provide us with good quality solutions in reasonable time, we also implemented an exact method using CP-SAT solver from OR-Tools, mostly to verify the optimality for small to medium instances. This Constraint Programming solution encodes the Travelling Tournament Problem as a set of Boolean decision variables and linear constraints to ensure that we satisfy the constraints of the problem.

3.2.1 What makes CP-SAT an Exact Method

1. CP-SAT methodically analyzes the solution space to ensure that no better solution exists while proving optimality.
2. The Branch-and-Bound Framework combines intelligent branching of binary variables with linear programming relaxations for bounding.
3. SAT-based Inference: Uses Boolean satisfiability approaches to remove infeasible regions from the search space.
4. CP-SAT's OPTIMAL status provides mathematical proof that the answer is optimal.

3.2.2 Complete Travel Cost Calculation:

Our Implementation employs a three-phase travel calculation that ensures accuracy:

- Home team returns: If a team scheduled to host a game is currently away, it must first return home.
- Away team travel: Teams travel from their current location to their opponents venue
- Final return home: All teams return to their home after the tournament ends.

3.2.3 Solver Configuration

- The CP-SAT solver is configured with a time limit to balance solution quality and computational resources. The parameters used are max_time_in_seconds, which controls the computational budget.
- The solver returns these status codes:

- OPTIMAL: indicates that a proven optimal solution is identified
- FEASIBLE: a feasible solution has been found, but optimality has not been established
- INFEASIBLE: there is no conceivable solution
- UNKNOWN: the time limit passed without finding a solution.

3.2.4 Schedule Extraction

Once a feasible solution is found, the program proceeds as follows:

- Reconstruct the tournament schedule from the values of $\mathrm{x}_{[}i, j, r]$
- Compute the total distance travelled by all teams
- Compute per-team travel statistics
- Show the output comprising of the full schedule per round, round-wise and total distance, Home/Away pattern for each team and the return trip costs.

3.3 Genetic Algorithm Approach

The Travelling Tournament Problem comprises of a large combinatorial search space that makes it hard for exact methods to be computationally feasible in providing a solution, especially for large instances.

Thus, to explore this search space efficiently and obtain good results within a reasonable time, we implemented a Genetic Algorithm. It begins with a consideration of a population of feasible solutions and keeps improving them iteratively, and does so with the help of evolutionary operators such as crossover and mutation. These operators are guided by a fitness function based on the total travel distance.

3.3.1 Chromosome Representation

Each chromosome has a full tournament schedule encoded in it as a list of rounds. Each of these rounds contain match pairs in the form of [home, away].

- Number of rounds $= 2 \times (n_{teams} - 1)$
- Each round covers all teams exactly once

3.3.2 Population Initialization

1. Generating a base feasible schedule using the circle method.
2. Randomly shuffling the rounds to create diverse candidates.
3. Validating each schedule before adding it to the population.

Population size = 50 in experiments.

3.3.3 Fitness Function

The fitness of a chromosome is the total travel distance for all teams.

- Travel is computed by tracking each team's location over time
- Travel cost is calculated when a team travels to the opponent's location from their current location or returns home

3.3.4 Genetic Operators

Crossover

- One-point crossover: select a random point, take rounds from p1 before and p2 after.

Mutation

- Two random rounds are swapped to introduce diversity.
- Applied with mutation rate = 0.25.

Selection

- New generation = valid offspring after crossover/mutation.
- If population $<$ required size, old individuals are reused.

3.3.5 Constraints Handling

We check the validity of each schedule to ensure:

- No team plays twice in the same round
- No consecutive matches against the same opponent
- Home or away streaks lie between L and U
- Each team faces every opponent once home and away

The validity check is done after every mutation and crossover before accepting a chromosome.

3.3.6 Termination Criteria

The Genetic Algorithm stops when either:

- generations limit is reached (150 in final runs)
- or computation time exceeds the max_time_sec i.e. 150 s

4 Results

4.1 Experimental Setup

- **Instances used:** NL4, NL6, NL8, NL10.
- **Parameters:** $L = 1$, $U = 3$ (as per dataset).

4.2 Performance Metrics

- Both algorithms give feasible solutions, even though it is not the optimal solution.
- The solutions have been checked by plotting the schedule, as it can be seen in figure 1 and figure 2

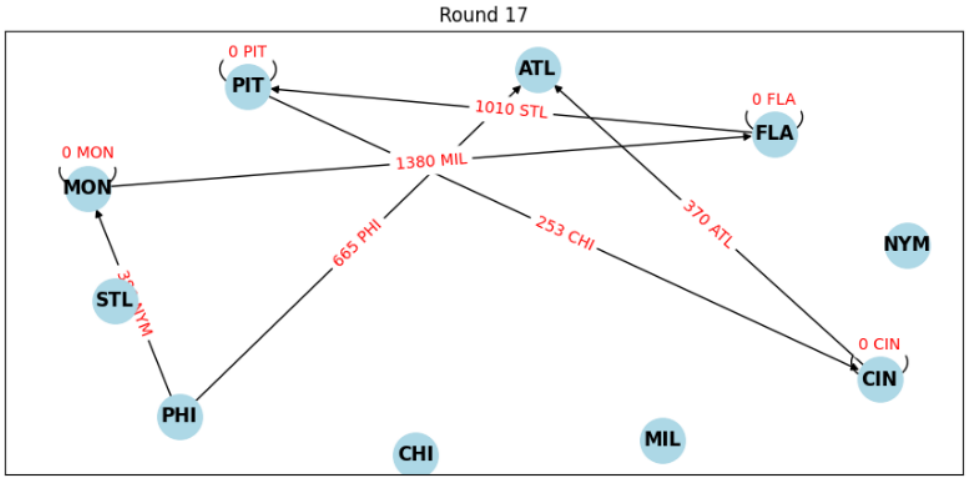

Figure 1: Example schedule for NL10 round 17.

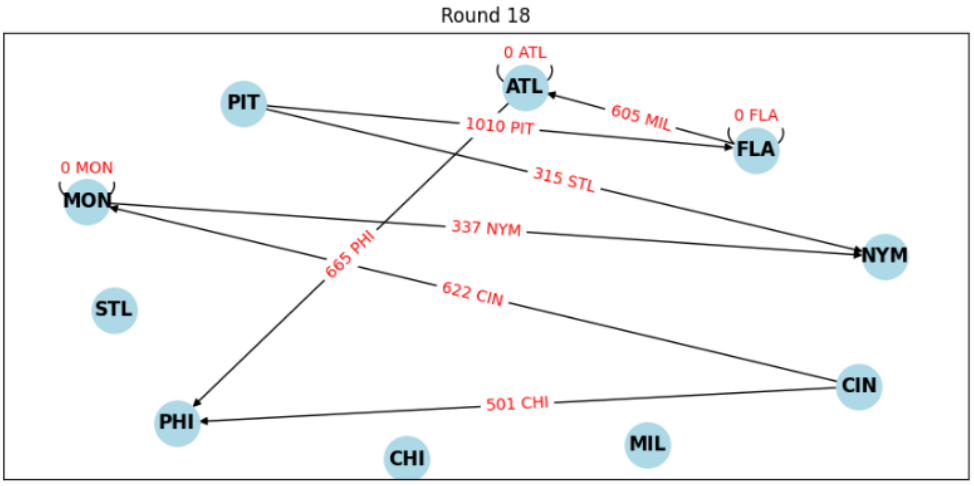

Figure 2: Example schedule for NL10 round 18.

4.3 Results Table

Below we have a table with the results of the Constraint Program algorithm (CP-SAT solver) and the Genetic Algorithm on multiple datasets. The Genetic Algorithm used the following fine tuning parameters:

- Generations: 150
- Mutation rate: 0.25
- Population size: 50

Instance	Teams	CP Travel	CP Time (s)	GA Travel	GA Time (s)	Gap (%)
NL4	4	9240	0.21	8559	0.58	7.95
NL6	6	34341	38.66	24766	1.17	38.66
NL8	8	60706	3000.15 (DNF)	43607	1.73	39.21
NL10	10	97918	3000.15 (DNF)	72125	2.55	35.76

Table 1: Comparison between CP and GA results across instances.

4.4 Schedules

Dataset: NL4

- Algorithm: Constraint Program 3
- Algorithm: Genetic Algorithm 4

Dataset: NL10

- Algorithm: Constraint Program 5
- Algorithm: Genetic Algorithm 6

4.5 Observations

- For small instances, the CP-SAT solver was able to quickly find an optimal solution (under 20 seconds), but times out for NL8 and NL10, which shows its problems with scalability. It also means that our CP solutions for those instances are not optimal, but what the solver was able to find before the timeout.
- The Genetic Algorithms solver was able to solve for all instances under 3 seconds, regardless of size.
- Our CP-SAT solver produced worse results than the GA solver, which suggests that our implementation may have been flawed.
- The gap between the solutions given by the algorithms remains steady at around 35-40%, the result from the NL4 dataset being an exception of this remark, which suggests that GA is a practical alternative when exact methods are not feasible to compute.

5 Group Member Contributions

- **Antal-Cionta, Ioan-Sorin**:
  - Conducted research on relevant literature to get familiar with the problem
  - Participate in every group meeting
  - Tried to come up with constraints when formulating our own model
  - Implemented visualizations in python such as graphs representing how each team moves every round and tables representing the schedule
  - Checked the validity of the solutions given by both algorithms
  - Contributed to filling in the results of the algorithms in the report
- **Chidambararajan Vijayalakshmi, Sibi Karthik**:
  - Studied relevant research papers to devise methods to model the problem and devised an aggregate list based on which *Literature Study* section was written
  - Contributed to formulating own constraints for the problem
  - Implemented the GA solver, which managed to arrive at optimal results after modeling of the problem
  - Helped to optimize the CP-SAT solver code to run within time constraints
  - Contributed to writing the *Literature Study* section of the report and checking the formatting and overall structure
  - Participated in all team meetings
- **Fushada, Jessica**:
  - Contributed to the understanding and formulation of the problem
  - Helped model the problem via brainstorm discussions and reading literature
  - Helped with the CP-SAT solver code and debugging, and validating results
  - Wrote the *Problem Formulation*, *Observations* and part of the *Literature Study* sections in the report and overall editing
  - Participated in all group meetings

- **Verma, Gourisha**:
  - Studied the problem and helped in formulating the initial mathematical model.
  - Contributed to discussions on decision variables, objective function, and constraints during group meetings and literature review sessions.
  - Contributed to formulating the ILP model, defining decision variables and constraints, and setting up the CP-SAT solver to obtain optimal solutions for smaller instances as well as GA solution design.
  - Drafting and structuring the *Solution Methods* section of the report.
  - Participated in all group meetings.
- **Vesalapu, Likhit**:
  - Implemented an exact solver (CP-SAT) using Google OR-Tools, managed to get desired output formats such as home/away pattern, detailed distance breakdown by rounds and final schedule.
  - Contributed in report writing for solution methods section (exact solver)
  - Find relevant research papers
  - Participated in every group meeting
- **Use of AI Tools**:
  - Used AI tools and LLMs as a supplementary tool for coding support, debugging assistance, and reviewing the overall report structure.

6 Appendix

	ATL	NYM	PHI	MON
0	@NYM	ATL	@MON	PHI
1	@PHI	@MON	ATL	NYM
2	MON	PHI	@NYM	@ATL
3	NYM	@ATL	MON	@PHI
4	PHI	MON	@ATL	@NYM
5	@MON	@PHI	NYM	ATL

Figure 3: Schedule for the NL4 dataset using ILP.

	ATL	NYM	PHI	MON
0	@PHI	MON	ATL	@NYM
1	MON	PHI	@NYM	@ATL
2	NYM	@ATL	MON	@PHI
3	PHI	@MON	@ATL	NYM
4	@NYM	ATL	@MON	PHI
5	@MON	@PHI	NYM	ATL

Figure 4: Schedule for the NL4 dataset using GA.

	ATL	NYM	PHI	MON	FLA	PIT	CIN	CHI	STL	MIL
0	NYM	@ATL	MIL	STL	CHI	CIN	@PIT	@FLA	@MON	@PHI
1	@PIT	MIL	STL	CHI	CIN	ATL	@FLA	@MON	@PHI	@NYM
2	@MON	PIT	FLA	ATL	@PHI	@NYM	@MIL	@STL	CHI	CIN
3	STL	CIN	PIT	FLA	@MON	@PHI	@NYM	@MIL	@ATL	CHI
4	@FLA	CHI	CIN	PIT	ATL	@MON	@PHI	@NYM	@MIL	STL
5	@CHI	@FLA	@MON	PHI	NYM	MIL	STL	ATL	@CIN	@PIT
6	@CIN	@PHI	NYM	MIL	STL	CHI	ATL	@PIT	@FLA	@MON
7	PHI	@MON	@ATL	NYM	MIL	STL	CHI	@CIN	@PIT	@FLA
8	MON	@PIT	@FLA	@ATL	PHI	NYM	MIL	STL	@CHI	@CIN
9	CHI	FLA	MON	@PHI	@NYM	@MIL	@STL	@ATL	CIN	PIT
10	MIL	STL	CHI	CIN	PIT	@FLA	@MON	@PHI	@NYM	@ATL
11	@STL	@CIN	@PIT	@FLA	MON	PHI	NYM	MIL	ATL	@CHI
12	PIT	@MIL	@STL	@CHI	@CIN	@ATL	FLA	MON	PHI	NYM
13	@NYM	ATL	@MIL	@STL	@CHI	@CIN	PIT	FLA	MON	PHI
14	FLA	@CHI	@CIN	@PIT	@ATL	MON	PHI	NYM	MIL	@STL
15	CIN	PHI	@NYM	@MIL	@STL	@CHI	@ATL	PIT	FLA	MON
16	@PHI	MON	ATL	@NYM	@MIL	@STL	@CHI	CIN	PIT	FLA
17	@MIL	@STL	@CHI	@CIN	@PIT	FLA	MON	PHI	NYM	ATL

Figure 5: Schedule for the NL10 dataset using ILP.

	ATL	NYM	PHI	MON	FLA	PIT	CIN	CHI	STL	MIL
0	@CHI	@STL	@FLA	MIL	PHI	CIN	@PIT	ATL	NYM	@MON
1	PIT	@PHI	NYM	STL	@CIN	@ATL	FLA	@MIL	@MON	CHI
2	MON	MIL	@PIT	@ATL	CHI	PHI	@STL	@FLA	CIN	@NYM
3	@CIN	@MON	FLA	NYM	@PHI	@MIL	ATL	STL	@CHI	PIT
4	CHI	PHI	@NYM	@PIT	@STL	MON	@MIL	@ATL	FLA	CIN
5	MIL	@CHI	STL	CIN	PIT	@FLA	@MON	NYM	@PHI	@ATL
6	@NYM	ATL	@CIN	@STL	@MIL	CHI	PHI	@PIT	MON	FLA
7	FLA	@CIN	@MON	PHI	@ATL	STL	NYM	MIL	@PIT	@CHI
8	@STL	CHI	PIT	@MIL	CIN	@PHI	@FLA	@NYM	ATL	MON
9	@MON	@PIT	CHI	ATL	MIL	NYM	STL	@PHI	@CIN	@FLA
10	CIN	MON	MIL	@NYM	@CHI	@STL	@ATL	FLA	PIT	@PHI
11	@MIL	@FLA	@STL	PIT	NYM	@MON	CHI	@CIN	PHI	ATL
12	@PHI	STL	ATL	@CHI	@PIT	FLA	MIL	MON	@NYM	@CIN
13	STL	@MIL	@CHI	@FLA	MON	@CIN	PIT	PHI	@ATL	NYM
14	@PIT	FLA	CIN	CHI	@NYM	ATL	@PHI	@MON	MIL	@STL
15	NYM	@ATL	@MIL	@CIN	STL	@CHI	MON	PIT	@FLA	PHI
16	PHI	CIN	@ATL	FLA	@MON	MIL	@NYM	@STL	CHI	@PIT
17	@FLA	PIT	MON	@PHI	ATL	@NYM	@CHI	CIN	@MIL	STL

Figure 6: Schedule for the NL10 dataset using GA.