A
Dissertation
On

A Novel Heuristic Approach to the Mirrored Travelling Tournament Problem

Submitted in partial Fulfilment of the requirement
For the award of the Degree of

Master of Technology

In

Computer Science & Engineering

Submitted By

Vipin Aggarwal

University Roll No. 2K11/CSE/21

Under the esteemed guidance of

Dr. Daya Gupta

HOD, Computer Engineering Department, DTU, Delhi

DELHI TECHNOLOGICAL UNIVERSITY

2011-2013

DELHI TECHNOLOGICAL UNIVERSITY

DELHI - 110042

CERTIFICATE

This is to certify that the dissertation titled "**A Novel Heuristic Approach to the Mirrored Travelling Tournament Problem**" is a bonafide record of work done at **Delhi Technological University** by **Vipin Aggarwal, Roll No. 2K11/CSE/21** for partial fulfilment of the requirements for degree of Master of Technology in Computer Science & Engineering. This project was carried out under my supervision and has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma to the best of our knowledge and belief.

Date: _____

(Dr. Daya Gupta)

HOD & Project Guide

Department of Computer Engineering

Delhi Technological University

ACKNOWLEDGEMENT

I would like to express our deepest gratitude to all the people who have supported and encouraged me during the course of this project without which, this work could not have been accomplished.

First of all, I am very grateful to my project supervisor Dr. Daya Gupta for providing the opportunity of carrying out this project under her guidance. I am deeply indebted to her for the support, advice and encouragement she provided without which the project could not have proceeded smoothly.

I am highly thankful to Ms. Lavika Goel research scholar in department of computer engineering, who enlightened me at every step of this project by giving helpful directions and guidance. I am grateful to all my friends and family for their continued support and encouragement throughout the research work.

Vipin Aggarwal

University Roll no: 2K11/CSE/21

M.Tech (Computer Science & Engineering)

Department of Computer Engineering

Delhi Technological University

Delhi - 110042

Abstract

This project aims at applying a new hybrid heuristic approach to the mirrored Traveling Tournament Problem (TTP). TTP is a tournament scheduling problem which abstracts the tournament structure of Major League Baseball. In this type of league, every team plays with every other team, once at its home and once away (at opponent's home). These type of tournaments are called Double Round Robin (DRR) tournaments. Aim of TTP is to make a schedule which incurs minimum travel distance for all the playing teams while conforming to some constraints like no team can play consecutively more than *n* matches home or away. While number of teams involved in standard benchmarks of TTP are no more than 32, still it is almost impossible to get an optimal schedule (with minimum travel cost) for even 10-team instance. TTP has been proved to be an NP-hard problem. Our aim is to apply hybridize Biogeography based optimization and Simulated Annealing heuristics to get *good* schedules of TTP. Simulated Annealing has shown its efficiency in tackling TTP benchmark instances with good results. However its very computation intensive technique and some instances take time even in number of days. Biogeography based optimization (BBO) is relatively new and fast heuristic approach which has not yet been applied to TTP. Although BBO itself is a global optimization technique, but due to its fundamental philosophy of sharing features among solutions, we cannot use it directly on TTP schedules because this can break the DRR structure of schedules and make them invalid. We use state of the art techniques to generate initial schedules fast. We modify these techniques so that their results can suitably be used by BBO. We use BBO as an intermediate step for fast convergence of solution to local optima. The best result produced by BBO is used as a starting point

for simulated annealing heuristic. It has been suggested in literature that the *goodness* of solutions produced by simulated annealing depends on its starting point. We aim at giving a good solution to simulated annealing to start with, in very less time. The performance of our hybrid approach is evaluated on standard National League and Brazilian soccer league benchmarks of TTP. The results are compared to the current best results. Our approach produces competitive results while consuming much lesser time as compared to best techniques available.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
Introduction

Professional sports tournaments is among one of the major economic activities around the world. In current era of globalization, major sports events happens in successions of one, two, three and four years around the world. Many countries bid for the rights to organize these events. These kind of events bring thousands of jobs and economic opportunities to their hosts. A major sports event like Olympic attracts large number of foreign tourists in form of spectators. They visit different places of hosting counties thus contributing to the incremental economic benefits. Millions of people around the world follow these sports events. They resorts to all those channels which can provide real time information about their teams' progress in each competition. Fans accesses sources like newspapers, radio, smartphone apps, television and Internet in their quest for information. Professional sport leagues like Major League Baseball (MLB), National Basketball Association (NBA), and Indian Premier League (IPL) etc. involve millions of fans. A huge investment is done in preparing players for competition. Sources of income for these leagues are TV and internet broadcasting rights, merchandising, stadium tickets. The accumulative figures for all these things easily reaches to billions of dollars for popular events.

Many challenging combinatorial optimization problems are inherent to these kind of leagues such as tournament scheduling, revenue maximization and logistic optimization. Television broadcasting rights of a popular sporting event brings in good proportion of the revenue earned from the event. For instance, In United States, $400 million is paid every year for the national telecast of baseball games and that much again is paid for local presentation [1]. Indian Premier League

garnered $ 1.6 billion as a 10 year contract of television broadcasting rights in four countries. The UK football team Manchester United is worth £400 and it receives £100 for overseas T.V. rights alone [1]. A good schedule can attract maximum television viewers, thereby increasing the revenue, while a poor schedule could be a reason of potential loss of income.

1.1 Motivation

Researchers of from a wide variety of fields such as operations research, scheduling theory, graph theory, Evolutionary computing, constraint programming, combinatorial optimization, and applied mathematics are being attracted to challenging problem of sports scheduling. This can be seen from the flurry of research work done in recent years. Example of recent research papers include Scheduling the Italian football league [2] , scheduling New Zealand basketball fixtures [3], County cricket timetabling using tabu search [4], Referee assignment is sports leagues (Dinitz [5], Duarte [6]) etc. For a well-documented and updated record on recent research in sports scheduling, see [7]. Because of the combinatorial explosive nature of problems involved in sports scheduling and management, variety of exact and approximate optimization techniques have been applied. These include integer programming (IP), constraint programming, metaheuristics, hybrid methods etc.

Team owners and other stakeholders in the league want to optimize their investments by playing on particular day and time in schedule at which they can get maximum viewers on T.V. and audience in stadium. Good fixtures (predetermined schedules) are important in order to have maximum revenue from tournament,

keep the interest of both media and fans and ensure the attractiveness of the games.

Schedules can even interfere (for the bad or the good) in the performance of players and can significantly affect the results and finances of every team participating in the tournament. Constraints related to finding the best schedule generally makes scheduling task very difficult. These constraints may arise due to the tournament structure, occurrence of festivals/holiday season in between the tournament, logistics, organizational, economical and fairness issues.

Among all the mathematical problems involved in sports, the general problem of scheduling the games is certainly the most studied area. It consists in determining the date, time and the venue on which each game will be played. Application of sports scheduling is common in popular league base sports such as basketball, baseball, football, cricket, and hockey, rugby etc. Lots of published work for scheduling of these sports can be found in literature [7]. However, there are also other relevant scheduling problems in sports. One of them famous problem is that of assigning referees to games. This problem also has multiple constrains and multiple objectives. Many interesting mathematical problems are closely related to these problems. Quadratic assignment problem is frequently encountered in tournament scheduling [8]. Graph theory has been traditionally used in scheduling sports events [9].

Whilst above facts are sufficient to motivate mathematicians and economists to do work in sports scheduling, our day to day involvement in sports motivated us to explore this interesting field. Even if someone is not very keen to follow sport leagues, one cannot avoid media coverage and madness associated with Olympic Games, Football world cups and other major tournaments. It certainly becomes interesting to understand how tournament organization is carried out behind the curtains.

Sports scheduling covers a wide area to be studied. It ranges from mathematics to computer science, operations research to Economics. For example, several researchers are interested in the underlying theoretical foundations (such as graph theory, Latin squares or factorizations) of the problems, or in real world applications of scheduling in different sports and leagues. We could also look at different types of problems that are faced by different competitions (such as maximizing gate receipts, scheduling fixtures and officials etc.). There are many facets of sports scheduling each one having its own interesting challenges and own unique benefits for the overall improvement of the game. Each individual and research who is attracted to solve interesting problems in sports scheduling has his/her own motivation.

The application of inexact methods to achieve unexpectedly good results for the NP-complete and NP-hard problems particularly motivated us to sports scheduling problems which are not only combinatorial explosive but are also so closely related to our day to day life.

Nature-Inspired Algorithms have been gaining much popularity in recent years due to the fact that many real-world optimization problems have become increasingly large, complex and dynamic. The size and complexity of the problems nowadays require the development of methods and solutions whose efficiency is measured by their ability to find acceptable results within a reasonable amount of time, rather than an ability to guarantee the optimal solution We have encountered a new upcoming metaheuristic named biogeography based optimization (BBO) which is defeating many veteran techniques like ACO, GA, PSO etc. for problems like TSP, path planning, land cover feature extraction etc. We were motivated to use it for a standard sports scheduling problem named traveling tournament problem (TTP) and see if its result could defeat

best published results in literature. To our amazement, we found combined BBO-SA technique quite competitive and highly efficient compared to best metaheuristics applied on TTP till date.

1.2 Problem Statement

TTP is a sports scheduling problem whose objective is to produce a double round-robin tournament which satisfies sophisticated feasibility constraints (e.g., no more than three away games in a row) and minimizes the total travel distances of the teams.

Optimizing team travel can be economically rewarding as the cost associated with air travel of a team crew along with their accommodation makes significant portion of expenditure spent on a tournament.

We will use a novel hybrid approach to solve the mirrored version of this NP-hard problem. We will exploiting the benefits of both conventional metaheuristics and nature inspired algorithms. We intend to use combined biogeography based optimization and simulated annealing to obtained least cost mirrored TTP schedules in minimum possible time. In one statement, our problem can be define as:

"To develop a novel heuristic by hybridizing biogeography based optimization and simulated annealing for solving mirrored traveling tournament problem"

1.3 Related work

Several algorithmic techniques have been applied to solve TTP. In her epic paper, Easton [1] framed the TTP problem formally using her sports scheduling

experiences with Major League baseball. Benchmarks instances for the TTP are available at [10]. These instances include abstract distance matrices obtained from National League called NL*n* instances, circular instances called CIRC*n* instances, and constant instances. Out of these three, first two were created by Easton [1] and the third one by Urrutia [11]. For constant instances, objective is to maximize total number of breaks, which is proved to be equivalent to minimizing total distance.

Easton *et al* [12] attempted to solve NL*n* instances of benchmarks using combined Integer programming and constraint programming. They used 20 processors and ran parallel code on them. Using these resources, 6-team instance was solved in few minutes while 8-team instance took 4 days of computation. It is still the largest instance which has been solved to optimality till date.

As soon as the problem was proposed, it became instantly popular among sports scheduling enthusiasts and number of attempts to solve TTP followed. Benoist *et al*. [13] combined Lagrange relaxation with constraint programming and developed a hierarchical architecture to attack TTP. Constraint programming is the main component of this architecture which captures the whole problem. Although problem could also be solved using CP only, but using some global constrains, bounds of the problem improved. Lagrangian relaxation provided these global constrains. Value of Lagrangian multipliers were modified by solving one sub-problem of each team. A sub-problem of TTP is a TSP problem for each team *i*.e. how to minimize the individual distance that one team travels in the tournament irrespective of other teams travel distance.

1.4 Scope of work

This project proposes a novel hybrid heuristic for the traveling tournament problem by hybridizing two metaheuristics: Biogeography based optimization (BBO) and Simulated annealing (SA). Simulated annealing has shown good results in the past as a standalone heuristic for TTP. BBO has recently been very popular metaheuristic and used for many combinatorial optimization problems, e.g. traveling salesman problem (TSP) [14], [15] . Experimental results proved competitive advantage of BBO over many popular metaheuristics.

As TTP is very closely related to TSP (in an optimal case, TTP contains *n* TSP tours, where *n* is number of teams in tournament), we try to exploit benefits offered by both metaheuristics. We implemented the hybrid algorithm and evaluated our results on publically available benchmarks at [10]. Results are compared with the best results known in literature.

Broadly scope of this work can be summarized as follows:

- To develop a fast constructive heuristic for developing good initial mTTP schedules.
- Adapt Biogeography based optimization to be used with fast constructive heuristic.
- Modify initial schedules using BBO and help in fast convergence to local optima. This step generates seed for simulated annealing.
- Use simulated annealing to optimize schedules.
- Test simulated annealing with various parameters and choose the final parameters empirically
- Evaluate our proposed approach on standard benchmarks for TTP

- Compare the results with the best results available in literature

1.5 Organization of the thesis

Remaining part of this thesis is organized in the following chapters:

Chapter 2: Problem Description & Literature Review

This chapter discusses the TTP problem in detail. It starts with illustrating the terminologies specific to TTP thus helping us to understand the problem. Then we present formal definition of TTP. All variants of this problem are also described with examples of schedule representation. We then study in detail all the noteworthy work that has been done on TTP till date.

Chapter 3: State of the art techniques for TTP

In this chapter, we explore the state-of-the-art techniques for TTP. We highlight their achievement and the shortcomings. The purpose of this chapter is to acquaintance ourselves with the latest developments in methodologies used for solving our concerned problem.

Chapter 4: Biogeography Based Optimization and Simulated Annealing

From the analysis of previous two chapters, we chose two metaheuristics which are suitable for our problem. This chapter explains these two metaheuristics while discussing all the minute details that are necessary to know, to adapt these algorithms for generic problem solving.

Chapter 5: Detailed System Architecture

This chapter explains what our system does. It is basically black box model of our system. We first explain the system's inputs and outputs in this chapter. Then we

explain the detailed system architecture along with the technical challenges faced in implementation. Block diagrams are resented for easy understanding of the system.

Chapter 6: Proposed Approach

This chapter illustrates our hybrid approach for solving mTTP. It describes the methods we choose to generate initial population of feasible mTTP schedules, the application of BBO and SA to improve them. We give BBO & SA algorithms adapted for mirrored TTP problem.

Chapter 7: Experiments and Results

This chapter illustrates the experimental setup used to obtain the results. All the data used for result generation is presented. Parameter values used for algorithm are discussed. Results of our hybrid approach are present and compared with best results in literature.

Chapter 8: Conclusion

We conclude our work in this chapter. Scope of future result is discussed. Challenges faced by our problem and where improved can be done is highlighted.

References

This section gives the reference details of sources used for studying and understating the problem and approaches used in this thesis.

Chapter 2
Problem Description & Literature Review

The Traveling Tournament Problem (TTP) represents the fundamental issues involved in creating a schedule for sports leagues where the amount of team travel is an issue. For many of these leagues, the scheduling problem includes a myriad of constraints based on thousands of games and hundreds of team idiosyncrasies that vary in their content and importance from year to year, but at its heart are two basic requirements. The first is a feasibility issue in that the home and away pattern must be sufficiently varied so as to avoid long home stands and road trips. The second is the goal of preventing excessive travel. For simplicity, we state this objective as minimize total travel distance.

2.1 TTP Terminologies

Single Round Robin Tournament (SRR): Every team plays with every other team once in complete tournament. In TTP, every team has to play with some team in each round. So for n teams, $n/2$ matches are played in each round. So if n teams are playing, then there will be total of n-1 rounds in which each team will be able to play against every other team.

Double Round Robin Tournament (DRR): Every team plays every other team twice in a complete tournament, once at home and once away (at opponent's home). This type of tournament is called a double round robin tournament. For n teams, a DRR tournament consists of $2n$-2 rounds.

Round Trip: The number of consecutive matches that a team plays outside of its home town is called its round trip. In TTP, round trip limit is set to 3.

Home stand: As opposed to Round trip, Home stand is the number of consecutive matches that a team plays at home. Its limit is also set to 3 in TTP.

In TTP, Distance matrix for 6-teams instance of TTP is shown in Table *I*. Row heading and column heading tells the city name, and numerical values represent distance between those cities.

Table 1:
Distance matrix of NL6 instance

	Team	*ATL*	*NYM*	*PHI*	*MON*	*FLA*	*PIT*
1	***ATL***	0	745	665	929	605	521
2	***NYM***	745	0	80	337	1090	315
3	***PHI***	665	80	0	380	1020	257
4	***MON***	929	337	380	0	1380	408
5	***FLA***	605	1090	1020	1380	0	1010
6	***PIT***	521	315	257	408	1010	0

2.2 Formal definition of TTP

An intuitive way of describing a problem is describing it in terms of its inputs and outputs. Following this approach, the TTP is defined as follows:

Input: A set of n teams $T = \{t_1,\ldots,t_n\}$ with n even; D a symmetric n by n integer distance matrix with elements d_{ij}; L, U are integer parameters.

Output: A double round robin tournament on the teams in T such that – The length of every home stand and road trip is between L and U inclusive, and – The total distance travelled by the teams is minimized.

For $U = n - 1$, the maximum value for u, a team may visit every opponent without returning home, which is equivalent to a traveling salesman tour. For small U, a team must return home often, and consequently, its travel distance increases. For U = 1, the objective becomes constant, and the problem is solely one of

feasibility. In practice, L = 1 and U = 3 or 2 are most commonly used. Additional background on these parameters and a description of other instances appear in [1].

2.3 TTP Constrains

At-most: No team can play more than 3 home/away games consecutively. In standard instances of TTP, L is taken as L and U as 3.

No-repeat: A game between T_i and T_j played at T_i's venue must not be followed by a game between T_i and T_j played at T_j's venue.

Mirrored structure: Second half of the schedule must be exactly replica of first half with the playing venues reversed. More formally, same teams play game with each other in round t and in round $t + (n\text{-}1)$ but with reversed venues. This constrain guarantee that No-repeat constraint will never be violated.

Mirrored tournament structure is commonly followed in Latin America. It was first formally defined by Urrutia [8]. An example of an mTTP is schedule is given in Figure 1.

	First Half					**Second Half**				
T\R	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***
1	-5	-6	3	-2	4	5	6	-3	2	-4
2	3	5	-4	1	6	-3	-5	4	-1	-6
3	-2	4	-1	6	5	2	-4	1	-6	-5
4	6	-3	2	5	-1	-6	3	-2	-5	1
5	1	-2	6	-4	-3	-1	2	-6	4	3
6	-4	1	-5	-3	-2	4	-1	5	3	2

Figure 1: Example of an mTTP schedule for 6 teams

A small variation proposed in TTP is TTP with predefined values define by Melo *et al.* [16]. In this variant, schedules are single round robin tournaments in which we pre decide the venue of each game to be played in tournament.

2.4 TTP Schedule Representation

We use Anagnostopoulos [17] representation of schedule. Column number represents round number. Row number represents a teams and value inside table in corresponding row represents its opponent team. Thus opponent of a team T_i in round k is given by value (i, k) in schedule. If (i, k) is positive, match is played at Ti's home else at its opponent's home. Each team starts its journey from its home site and travels to its destined venue as given in schedule. After the last round, team returns to its home site (if last game was away). Travel cost of team 1 as per schedule given in Fig 1 is:

dist ($T_{1)} = d_{15} + d_{56} + d_{61} + d_{12} + d_{21} + d_{13} + d_{31} + d_{14} + d_{41}$

Here d_{ij} represents the distance between home venues of teams T_i and T_j. Total cost of schedule is calculated as sum of the distances that each team travel to play all of its matches. Thus cost of schedule will be as follows:

$$\text{Cost (S)} = \sum_{i=1}^{n} dist(T_i)$$

2.5 Variants of TTP

A number of researchers have developed variants of the Traveling Tournament Problem. This section contains a list of those variants.

2.5.1 Non round robin scheduling

This problem is a variant of the Traveling Tournament Problem proposed by Douglas Moody. In this variant, teams do not play a double round robin tournament but rather there is a "Matchups" value between teams *i* and *j*, which gives the number

of times *i* must visit *j*. The (regular) TTP is a Non-RR TTP with a matchup value of 1 for all *i* not equal to *j*.

2.5.2 Relaxed TTP

This problem is a variant of the Traveling Tournament Problem and is proposed by Renjun Bao and Michael Trick. In this variant, the schedule is not compact: teams have byes in their schedule. The number of byes is controlled by a parameter *K*, the number of byes per team in the schedule. *K*=0 corresponds to the normal TTP.

Byes are ignored in determining the length of a homestand or roadtrip, and in determining whether a repeater has occurred.

2.6 Literature Review

TTP is quite a popular problem in fields of operations research and Evolutionary computing. It's an optimization problem very closely related to traveling salesman problem which raises interest in every research as she found herself familiar with the problem even looking it first time. Many successful attempts have been recorded in literature solving TTP with good results. We discuss some of the most prominent of those here in this section.

Using Simulated Annealing Anagnostopoulos *et al.* [17] could beat all the previous best results (at the time of writing his paper) of TTP. He used complex neighbourhood moves to explore a large solution space. He explored both feasible and infeasible reasons and this decision translated in to better results. Strategic oscillation and reheats were used to avoid local optima. His work is still citied by most authors because of the path breaking results he achieved with a very simple approach using a simple algorithm.

Costa *et al*. [18] proposed an iterated local search (ILS) heuristic for TTP with predefined values. In their approach they used two types of perturbations and two local moves to explore neighbourhood. They did not use well known polygon method that most other researchers use as a tool to generate initial solutions. Instead they resorted to canonical-1 factorization method. This method was applied on sub-graphs and main graph of tournament to get initial feasible solutions. With their numerical results they proved ILS is much better than integer programming.

Ribeiro *et al* [8] proposed a hybrid heuristic based on the principles of GRASP (Greedy randomized adaptive search procedure) and ILS metaheuristics for mirrored TTP (mTTP). For generating initial schedules, they proposed a fast constructive heuristic using polygon method. This method first created an abstract schedule based on an initial random permutation. Then this abstract schedule is converted to a real schedule by mapping abstract teams to real teams considering distances among the cities of real teams. This helped in improving the schedule. After that, stadiums are assigned to teams by trying to keep road trips to maximum length. Obtained schedules were of much better quality than those of random schedules. They used four different neighbour structures to be used with ILS and GRASP. Some of them were complex, so ejection chains had to be used with them. Their results were competitive with best results available at the time of writing and took at most 15 minutes for the largest instance of problem. Some results of this hybrid result even beat the best results and marked themselves as new standards to be challenged.

Urrutia *et al.* [19] again came back with their new methodology to propose new lower bounds for the TTP proposed by Easton *et al.* [1]. He considered the difference between minimum numbers of road trips that each team needs to travel

for completing the tournament with the optimal solutions which were known before hand for equivalent size constant instances.

The work of Gaspero *et al.* [20] is also among the most cited heuristic work for TTP. He used tabu search to generate approximate solutions for TTP. He made use of complex neighbourhood structures and combined them to explore systematically a large solution space to reach competitive solutions. He compared the results with best results mentioned in literature and improved upon some instances.

There have been previous attempts to hybridize simulated annealing (SA) with other heuristics due to the path breaking results that SA achieved. One such work much appreciated in literature is by Lim *et al.* [21]. In his work, he proposed a hybrid algorithm by combining simulated annealing with hill-climbing for TTP. The search space is divided into a team assignment space and a timetable space. The team assignment space is explored by hill-climbing algorithm while simulated annealing explores the timetable space. A controller assigns teams to game slots and calls the simulated annealing component. Simulated annealing component helps in generating better timetables. Now the timetable with best schedule (in terms of lowest travel cost) is transferred to the hill-climbing component for further processing, which tries to improve schedule by better team assignments. This module returns those mapping of team assignments to simulated annealing component which generates the best schedule. This process of refinement continues until the schedules do not stop improving for a specified fixed number of consecutive iterations or when a predefined time limit is reached. The whole idea of this hybrid approach boils down to this: Try to improve team assignments if and only if their

associated timetables have a higher chance of giving better schedules thereby saving unnecessary computation.

While discussing the remarkable work done for TTP, it becomes absolutely essential to discuss the approach taken by Hentenryck *et al.* [22]. It is because most of the current best results for mirrored TTP that no one could challenge till date were achieved by Hentenryck. He improved upon the approach taken by Anagnostopoulos *et al.* [17]. He modified the objective function considering mirrored constraint as a soft constraint, and kept a penalty for its violation. This modification helped in exploring a large solution space while making it easier to return to feasible region once the search goes in to infeasible space. His results improved upon most of the best results which existed at the time of writing.

Cheung KKH [23] proposed a two-phase method based on generating timetables from 1- factorizations and finding optimal home/away assignments solves the mirrored traveling tournament problem benchmark instances NL8 and CIRC8.

In this chapter, we gave a detailed literature survey which throws light on the various efforts that have been done to solve TTP efficiently till date. Now let us discuss the various mathematical techniques and metaheuristics that produced exceptional and record breaking results when they were first applied to TTP. We try to find state of the art technologies so that we can put efforts in right direction by not using outdated idea or technology.

Chapter 3
State of the art techniques for TTP

This chapter contains the brief description of most successful and state of the art techniques which have been applied to solve TTP generating good results. The work on TTP started in 2001. In past 12 years, there have been numerous attempts to lower the bounds of this NP-hard problem. Many have achieved remarkable results for small instances but tackling the large instance problems (team size 12 or more) is still a big challenge.

We can broadly classify the approaches, which have been used to solve TTP so far, in to two categories:

1. Conventional metaheuristics
2. Nature inspired Algorithms

In following sections, we explain these approaches and define the state-of-the-art techniques which have been successfully applied on TTP

3.1 Conventional metaheuristics

These techniques include the approximate algorithms which have been around for quite some time and are either originated from mathematics models directly (constraint programming, integer programming, tabu search) or by inspiration from other engineering principle, like simulated annealing emerged using idea of metallurgy process. These methods have been applied to a wide range of problems and lots of theoretical work can be found on them. Below we given few of them which when introduced for TTP showed record breaking results.

3.1.1 Combined integer and constraint programming

This was the first solution posted for TTP by its creators Easton et al. [12]. Their solution methodology for the TTP is a branch-and-price (column generation) algorithm in which individual team tours are the columns. In branch and price, the linear programming (LP) relaxation at the root node of the branch and bound tree includes only a small subset of the columns. To check the LP objective, a sub problem, called a pricing problem, is solved to determine whether there are any additional columns available to enter the basis. If the pricing problem returns one or more columns, the LP is re-optimized. If no more columns can be found to enter the basis and the LP solution is fractional, the algorithm branches.

Branch-and-price is a generalization of branch-and-bound with LP relaxations. In their combined integer programming-constraint programming approach, they used constraint programming to solve the pricing problem.

Shortcomings of Combined Constrained and integer programming approach

Instances with $n = 4$ were nearly trivial to solve. Instances with $n = 6$ are more challenging. They explored several models that can solve these instances in a reasonable amount of time without parallel programming. When 20 processors are used to solve instances with $n = 6$, the computation time is on the order of minutes. Finally, they found it is necessary to use parallel programming to solve instances with $n = 8$ teams. On 20 processors, these problems take approximately 4 days.

Thus it is evident that the computation time needed by this approach is enormous even for small instances.

3.1.2 Tabu Search

Tabu search reignited the research in TTP in 2006 when Di Gaspero [20] used it with recording breaking results at their time of writing. In their work, a family of tabu search solvers for the approximate solution of the TTP is proposed. They make use of complex combinations of many neighborhood structures. The different neighborhoods are thoroughly analyzed and experimentally compared. The solvers are evaluated on three sets of available benchmarks and their outcomes are compared with previous results presented in the literature.

Shortcomings of tabu search

Di Gaspero found that due to the exhaustive exploration of the neighborhood, tabu search is intrinsically much slower than simulated annealing to perform each single iteration. In order to be competitive with the other techniques, tabu search needs to be implemented efficiently. In their case, many points still need to be improved, especially regarding the computation of the cost difference of two neighbor states

3.1.3 Simulated Annealing

Proposed by Anagnostopoulos [17] and further improved in [22], simulated annealing proved to be quite successful in solving TTP. Anagnostopoulos proposed a hybrid algorithm for the TTP is proposed, based on the simulated annealing metaheuristic and exploring both feasible and infeasible schedules. The heuristic buys some principles from other metaheuristics: it uses a large neighborhood with complex moves and includes advanced techniques such as strategic oscillation and reheats to balance the exploration of the feasible and infeasible regions and to escape local minima at very low temperatures. It matches the best-known solutions on the small instances and produces significant improvements over previous approaches on

the larger instances. The algorithm is claimed to be robust, because the worst solution value it produced over 50 runs is always smaller than or equal to the best known solutions.

Shortcomings of Simulated Annealing

Although most of the best results present in literature for TTP and its variants are produced by simulated annealing approach, but the main drawback in this approach lies in the computation time it takes. The computation requirement of this technique makes it infeasible to be used as real life problem solving technique. It took around 54 hours *i*.e. more than 2 days to find good results for 16 team instance of TTP using SA.

3.1.4 Iterated Local Search

This approach was used by Urrutia *et al.* [8]. In their work, a hybrid heuristic combining principles from the GRASP and ILS metaheuristics is proposed for the mirrored TTP. A three-step constructive heuristic is used to build good initial solutions. In the first step, the canonical 1-factorization is used for constructing a timetable with placeholders. Next, a greedy heuristic is used to assign teams to placeholders. The venues of the games are set round by round and local search is used to repair possible infeasibilities in the last step of the constructive heuristic. The hybrid heuristic makes use of four simple neighborhoods for local search and one ejection chain neighborhood for perturbations. The results obtained by the hybrid heuristic were even better than the best known at the time of writing for some instances of the less constrained TTP, with execution times limited to 15 min. State-of-the-art algorithms at the time of writing usually reported up to several days of computation time. It is also shown that the constructive algorithm is very quick and

produces good initial solutions that improve the quality of the best solution found by the hybrid heuristic.

Shortcomings of Iterated Local Search

This technique was able to produce results reasonably quickly than previous known techniques for TTP but again trade off lies here in the cost of schedules which it generates. The maximum difference between the cost of its best schedule and best known schedule in literature reached 17%. This is unacceptable as more scope of improvement lies in cost optimization which was main objective of TTP.

3.2 Nature inspired algorithms:

These methods have been recently discovered and are based on various phenomenon in nature that makes organisms find their food (Ant colony, bee colony), help in existence and evolution (biogeography, genetics), develop social behaviours (bird flocking, fish schooling) etc. Although these algorithms have shown remarkably good results on numerous optimization problems, much less theoretical work has been published on them compared to conventional metaheuristics. Nevertheless, they achieve the objective very well which is to find a near optimal solution for a given problem. In sections given below, we discuss few problem which have been applied on TTP and their results have shown noteworthy improvement over results that were existing in literature.

3.2.1 Genetic Algorithm

Evolutionary computation was first used by Biajoli *et al.* [24] to solve TTP. The methodology used to solve the problem is based on the use of Genetic Algorithms in association with the metaheuristic Simulated Annealing. The idea is to use the

Genetic Algorithm as construction phase, generating new solutions starting from the individuals' crossing and the Simulated Annealing to improve the local search in those new solutions.

In this work, Genetic Algorithm was implemented that uses the Simulated Annealing metaheuristic to address new individuals to a local optimum. The application of local search in the individuals can be related with the combination of learning and evolution. In general, the learning is a search for the near viable solution and the modifications will be incorporate for the individual. The use of the SA metaheuristic leaves the stage of local search more aggressive, resulting in individuals more and more adapted inside of the population. A compact representation of the chromosomes (individuals) was proposed for the application of the GA. The chromosomes are submitted to an algorithm of code expansion, which decodes them in scales of games.

Shortcomings of Genetic Algorithm

This approach for TTP was inconsistent throughout out the standard problem instances. For some instances, it produced very good results, for NL10, the gap percentage reached up to 12% showing poor quality of solution generated. Thus it proved to be unreliable technique for real life problem solving.

3.2.2 Particle Swarm Optimization

Alireza Tajbakhshı et al. [25] proposed a hybrid PSO-SA approach for TTP. In the proposed algorithm, two metaheuristic methods are used: Particle Swarm Optimization (PSO) and Simulated Annealing (SA). This hybrid algorithm applies 0-1 version of PSO in the first phase and generates many schedules rapidly. In the second phase of the hybrid algorithm, an SA approach applies the best schedules

achieved in the first phase as initial schedules and improves them. The proposed algorithm leads to an optimal solution for the National League (NLn) instances of the TTP with 4, 6, and 8 teams.

Shortcomings of PSO

Work of Tajbakshi provided an alternative mathematical model for TTP (the first was given by Easton). Their results where comparable to most results in literature with a new approach but still they could not beat the heavy computation involved for TTP schedules generation. They took 7200 seconds, *i*.e. 2 hour for generating results for smallest of the instances.

We can conclude from this chapter that in recent years, a trend of applying nature inspired algorithms to TTP is gaining popularity. This is due to their exceptional abilities in producing near optimal results in much lesser time than conventional techniques. Now let us explore various nature inspired algorithms and choose the best fit for our problem.

3.2.3 Ant Colony Optimization

In Ant Colony Optimization (ACO), a set of software agents called artificial ants search for good solutions to a given optimization problem. To apply ACO, the optimization problem is transformed into the problem of finding the best path on a weighted graph. The artificial ants (hereafter ants) incrementally build solutions by moving on the graph. The solution construction process is stochastic and is biased by a pheromone model, that is, a set of parameters associated with graph components (either nodes or edges) whose values are modified at runtime by the ants.

Chen *et al.* [26] used an ant based hyper heuristic for solving TTP. In their proposed model they constructed a network in which every vertex represents a low-level heuristic. A number of ants, each of which represents a hyper-heuristic agent, are located uniformly among the vertices of the network and carry initial solutions. Each ant traverse particular edges and reach the next vertex. Once an ant arrives at a new vertex it applies the low-level heuristic at that node. They allowed the ants to visit the same node many times. Indeed, ants can cycle back to the same vertex so that the same heuristic is repeatedly applied to the current solution. In addition to the pheromone trails, they used the concept of 'visibility'. Visibility represents how quickly the heuristic at a potential vertex takes to compute. This is on the assumption that short, good quality heuristics are to be preferred to good quality heuristics that take a long time to compute.

Shortcomings of ACO

1. Theoretical analysis is difficult
2. Sequences of random decisions (not independent)
3. Probability distribution changes by iteration
4. Research is experimental rather than theoretical
5. Time to convergence uncertain

Although using ACO as a hyper heuristic, Cheng *et al.* could find solutions for all NL*n* instances of TTP, but their results were far from optimal. They were even worse than results available at their time of writing. With ACO, they could just invent an approach suitable enough to give results (good or poor) for all instances.

In this chapter, we explored various techniques that have been applied on TTP and have produced good results. While studying various nature inspired techniques, we

came across BBO as a general metaheuristic to optimize combinatorial problems with good success rate [14], [15]. We will explore BBO and simulated annealing in detail in next chapter. We'll try to study all minute details of these two metaheuristics so that we can adapt them to our concerned problem. Then we will go ahead and hybridize these two techniques and will evaluate our hybridized approach on publically available standard benchmarks.

Chapter 4
Biogeography Based Optimization and Simulated Annealing

This chapter contains the detailed discussion of two metaheuristics that we used for solving TTP: Biogeography bases optimization and Simulated Annealing. We will study their origin, algorithmic techniques, their strengths and weakness and compare them with related algorithms. We will first start off with BBO and then explain simulated annealing.

4.1 Biogeography Based Optimization

4.1.1 Background

Scientific work to understand biogeography started long back in 19th century. But the first mathematical model describing biogeography only came out after the revolutionary work of Robert MacArthur and Edward Wilson in 1967. They were mainly focused on how species distribute among neighboring islands, how and why species migrate, reasons of their evolution and extinction. Although there had been quite an enthusiasm after research work of MacArthur and Wilson in field of biogeography, no computer science papers taking advantage of biogeography were ever published till 2008.

In his seminal paper in 2008, Dan Simon [27] used the mathematical models developed by MacArthur and Wilson and invented a new metaheuristics called Biogeography-Based Optimization (BBO). Since its inception, there have been various improvement proposed to improve BBO ([28], [29], [30], [31]). Simon and his colleagues have published lots of research work to prove the competiveness of

BBO with other main stream metaheuristics like genetic algorithm, ant colony optimization etc. with mathematical theory and experimental results.

4.1.2 Biogeography

In the original biogeography models developed by MacArthur and Wilson, term island was used to define a *habitat* geographically isolated from other habitats. Simon instead used the term habitat in place of island considering its more generic meaning and contextual accuracy. Geographical areas that are well suitable to live and evolve for animals have high suitability index (HSI). Features that are responsible for HSI includes rainfall, biological diversity, vegetation, temperature, land area etc. Variables that represents these features are effects HSI are called suitability index variables (SIVs). SIVs are independent variables, while HSI can be considered as a dependent variable.

Habitats having high HSI tends to have more number of species while with low HSI will have less count of species. If number of species on a habit rises, some migrates to nearby habitats. It is not that all the members of an emigrating specie disappears from the previous home. Just few representative members move. So emigrating specie has its existence on both emigrating and home habitats. Thus high HSI habitats have high emigration rate. Now as the number of species are high on these habitats, there is little space of new species that might come from neighboring habitats. So high HSI habitats tends to show low immigration rate and therefore high HSI habitats are resistant to change because of very less probability of accepting new species.

Contrary to that, low HSI habitats have high immigration rate due to their sparse population. The species which tries to immigrate to low HSI islands have

better chances to find space due to less competition. As the population grows on low HSI habitats, their HSI tends to increase as suitability of a habitat is directly proportional to its biological diversity. But in case, HSI of a habitat remains low for long time, species on it are likely to go extinct which further open the doors for immigration by reducing competition. These facts make us believe that low HSI habitats are more dynamic in species distribution.

Relating biogeography to problem solving

Now let's come to the point how nature's way of species distributing species can help us in problem solving. Suppose we have a problem and some of its possible solutions. Problem can be of any field (sports, business, urban planning, engineering, science etc.). For biogeography concepts to apply on a problem, there must exist some quantifiable mechanism to measure the suitability of a solution. Relating problem to our biogeography model, a good solution is analogous to a habit with high HSI and a bad solution represents a habit with low HSI. High HSI solution tends to be more resistant to change and they won't easily change until some better candidate solutions emerge in the population. Low HSI solutions will get some *traits* of high HSI solutions to improve themselves. This addition of new features may rise the HSI of poor solutions thereby improving their quality probability of acceptance as final solution. This approach of problem solving was christened as Biogeography-Based optimization (BBO) by Simon [27].

Similarities and differences of BBO with PSO and GA

BBO share its problem solving attributes with many popular biology based metaheuristics. Like particle swarm optimization (PSO) and genetic algorithm (GA), BBO share features among its candidate solutions. In GA, solutions 'die' at the end of each generation while BBO and PSO solutions never actually 'die' although they

may modify as optimization process progress. PSO solutions may form groups based on similarities and this may have a significant impact on how features are shared. But GA and BBO algorithms do not put any mandatory requirement to form cluster of solutions, although they do not restrict it too.

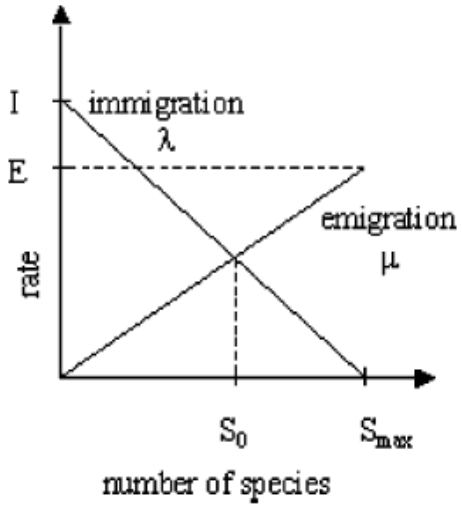

Figure 2: Species model of a single habitat

Figure 2 illustrates how species move in a habitat based on their numbers. Emigration rate μ and immigration rate λ are functions of number of species present in the habitat. We will explain the different parts of curve shown above one by one.

Highlights of Immigration curve

- Maximum possible immigration rate for a habitat is *I*.
- Value of immigration rate is *I* only when number of species in a habitat is zero.
- Immigration rate decreases as number of species increases due to crowded space and competition.
- When the number of species reaches to its maximum value that a habitat can support, immigration rate reduces to zero.

Highlights of emigration curve

- If no species exist in habitat then emigration rate becomes zero.
- Emigration rate increases with increase in population of species as more of them are able to leave their habitat to explore other habitats.
- There is limit to the emigration rate which is given by E. It happens when a habitat contains maximum number of species which it can support.

The state of Equilibrium

- S_0 is the equilibrium number of species. When count reaches to S_0, immigration and emigration rate becomes equal.
- Every habitat eventually reaches to the state of equilibrium after a particular period.
- Curve may deviate from point of equilibrium once it reaches there because of following two reasons.
    - *Positive excursions*: It could be due to sudden of speciation or due to sudden spurt of immigration because of some catastrophic event on neighboring island.
    - *Negative excursion*: It may happen due to epidemic or due to introduction of some ravenous predator.

4.1.3 Operation in BBO

The two operations that are fundamental to BBO are migration and mutation [27]. Both operations with their problem independent algorithms are explained below.

1. Migration

Emigration and immigration constitutes the migration process. They are actually inversely proportional to each other. For clarification see Figure 3. It is illustrated

there that we have two candidate solutions one bad (S1) and one good (S2). Number of species on habitat S1 are lower than that of number of species on S_2. Fitness or HSI of a solution is directly translated to the number of species it contains. So the poor one (S_1) will have higher immigration rate than better one (S_2) *i*.e. $\lambda_1 > \lambda_2$ and will have lower emigration rate than better one *i*.e. $\mu_1 < \mu_2$.

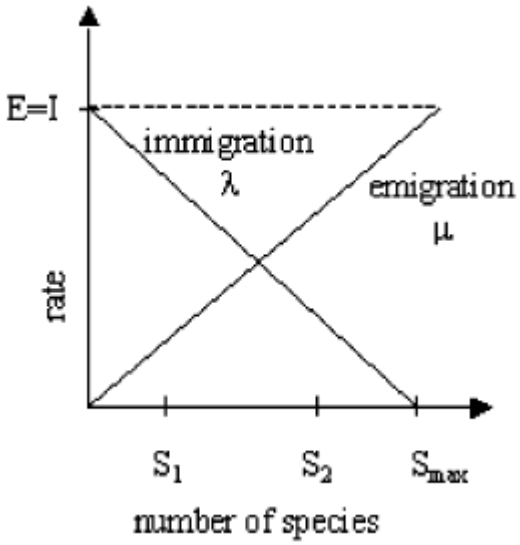

Figure 3: Comparison of two candidate solutions based on their λ and μ

So we come to the conclusion that if we have fitness of a solution, we can calculate its number of species relative to other solutions. And we have number of species, we can calculate λ and μ.

In BBO, we modify each habitat with a probability of P_{mod}. If a habitat H_i is selected to be modified, the immigration rate λ_i probabilistically decides among all the SIVs of a habitat, which ones will be modified. If some SIV is selected to be modified from habitat H_i, then emigration rate μ of all other habitats probabilistically decides which habitat will give its SIV (selected randomly or using some problem dependent predefined rule) to H_i. A general algorithm of migration is given below.

```
Select H_i with probability ∝ λ_i
If H_i is selected
    For j = 1 to n
        Select H_j with probability ∝ μ_i
        If H_j is selected
            Randomly select an SIV σ from H_j
            Replace a random SIV in H_i with σ
        end
    end
end
```

In BBO, as with other population based metaheuristics, we use concept of elitism to prevent best solutions from getting corrupted by migration process. By keeping probability of modification, P_{mod} to zero for best solutions, we can prevent them from modification.

2. Mutation

Habitats can drastically change by random cataclysmic events. These events are rare but they do happen and must be reflected in BBO model. These events causes the state of a habitat to shift from its equilibrium. Mutation is the process by which cataclysmic events are reflected in BBO. Probability of mutation is kept to a very small value for the fact that these events are very rare. By looking at Figure 3, we analyse that probability of occurrence of very low number of species and very high number of species is fairly low. Natural habitats are likely to evolve over time steadily and gradually. Among any set of habitats, most of them have number of species with in some range of equilibrium. This state is achieved after a long time since the habitat came in to existence. Thus at any given point in time, each population member has an associated probability which tells that it was expected as a solution. If a habitat H_i with probability P_s has currently very low number of

species or very high number of species then it is surprising that it exists. We assume that it is due to some cataclysmic event and thus we like to mutate it. So the probability of mutation is high for very good and very poor solutions and low for mediocre solutions. Mutation probability is guided by the equation

$$m(H) = m_{max}\left(\frac{1 - P_H}{P_{max}}\right) \qquad (1)$$

Here *m(H)* is the mutation probability, m_{max} is the user defined parameter (maximum value that mutation can take. $0 \leq m_{max} \leq 1$. P_{max} is the apriori probability of occurrence of the best habitat, while P_H is the apriori probability of occurrence of habitat H. The whole motive of mutation is to increase diversity in population. Without this, high HIS solutions tend to be dominant. Mutation give chance of improving to low HSI solutions. Good solutions also have chance to improve further using mutation. So mutation scheme is applicable to good and poor solutions equally. Average solutions may have been improving already. So they are less likely to mutate. Although with this approach best solutions of our population may get corrupted. But as with migration, we can set elitism approach here to prevent corruption of few best solutions. Mutation concept is common to GA, and all different types of mutation that are applicable in GA can be used here also.

Mutation general algorithm is given below.

For $j = 1$ to m
  Use λ_i and μ_i to compute the probability P_i
  Select SIV $H_i(j)$ with probability $\propto P_i$
  If $H_i(j)$ is selected
    Replace $H_i(j)$ with a randomly generated SIV
  end
end

4.1.4 BBO Algorithm

Working of BBO can be summarized in following sequence of steps [27]

1. Initialize parameters of BBO. It includes:
    a. Mapping problem's solutions to SIVs and habitats.
    b. Initialize maximum species count $S_{max,}$ maximum immigration rate. $\lambda_{max,}$ maximum emigration rate μ_{max} and maximum species count S_{max}
2. Initialize a random set of habitats (solutions) where each habitat corresponds to a quantifiable solution of given problem.
3. For each habitat, calculate λ and μ using its species count.
4. Use immigration and emigration rates to probabilistically modify each non-elite member of habitats.
5. Re-compute HSI of habitats. Update species count using HSI. Then mutate each non-elite member using (1).
6. Go to step 3. This loop will be terminated if either predetermined number of iterations have reached or required quality solution has been found.

It is to be noted here that after modification of a habitat after migration and mutation operations, it might become infeasible according to solution constraints of problem. So feasibility must be verified of each solution in every iteration.

4.2 Simulated Annealing

Simulated annealing (SA) is a random search process which is analogous to the annealing process used in metallurgy. In metallurgy, heating and cooling to a metal

is done in a controlled way to minimize the energy of its molecules and convert it into a uniform crystalline structure with minimum defects. In the same way, in SA, slow cooling is imitated by slowly decreasing the probability of accepting worse solutions during the search process.

SA is used to find an approximate global optimum for a problem which has discrete and large search space. It is proved to be better than exhaustive enumeration when search space is huge objective is to find near optimal solution rather than optimal solution.

4.2.1 Background

SA was first introduced by Kirkpatrick *et al.* [32] in 1983 to deal with nonlinear problems. Approach of SA for global maximization is similar to that of a bouncing ball which bounces over mountains (local optima) from valley to valley. The process begins at a high "temperature" which powers the ball to take very high bounce and allow it be in valleys surrounding mountains (search exploration region). As the temperature declines the bouncing power of ball decreases and the range of valleys it can explore. But due it slow cooling, we hope that SA has crossed all local optima that might have been appeared in its search space. A generating distribution is a set in SA which gives a valley or states to be explored. Similar to that is set called acceptance distribution. Based on the differences in value of present valley being explored and the last explored valley, it probabilistically decides whether to stay in the present valley or bounce over it to explore some other. Temperature controls the generating distribution and acceptance distribution. It has been proved mathematically that SA can find the global optima by carefully controlling the temperature. However, this requires infinite time.

4.2.2 The method

Simulated annealing's exceptional ability to avoid local minima sets it apart from other rival neighborhood based metaheuristics. If the aim of SA's objective function f is to minimize some solution, then SA not only accept changes that minimize it, but also those changes that increases its value. Although the acceptance of latter is associated with some probability shown by (2).

$$P = \exp(-\delta f/T) \qquad (2)$$

where δf is the non-desired change in function value and T is a control parameter better known as *"temperature*". It works independent of the objective function. The implementation of the basic SA algorithm is straightforward. Flow chart showing basic working of SA is shown in Figure 4.

Following are essential elements of Simulated Annealing:

1. Representation of possible solution of problem
2. A generator function which modify solutions to generate new ones.
3. A fitness/cost evaluation function which gives measure to compare solutions
4. An annealing schedule: It contains an initial temperature at which search starts and rules to modify that temperature as search progresses.

4.2.3 Strengths and Weakness of SA

Strengths

- SA can deal with noisy and chaotic data with many constraints, it can computer near optimal solutions of highly non-linear models efficiently.
- Compared to other local search methods, if offers flexibility in reaching to global optimal solutions.
- Versatile in nature as does not have any restrictive properties

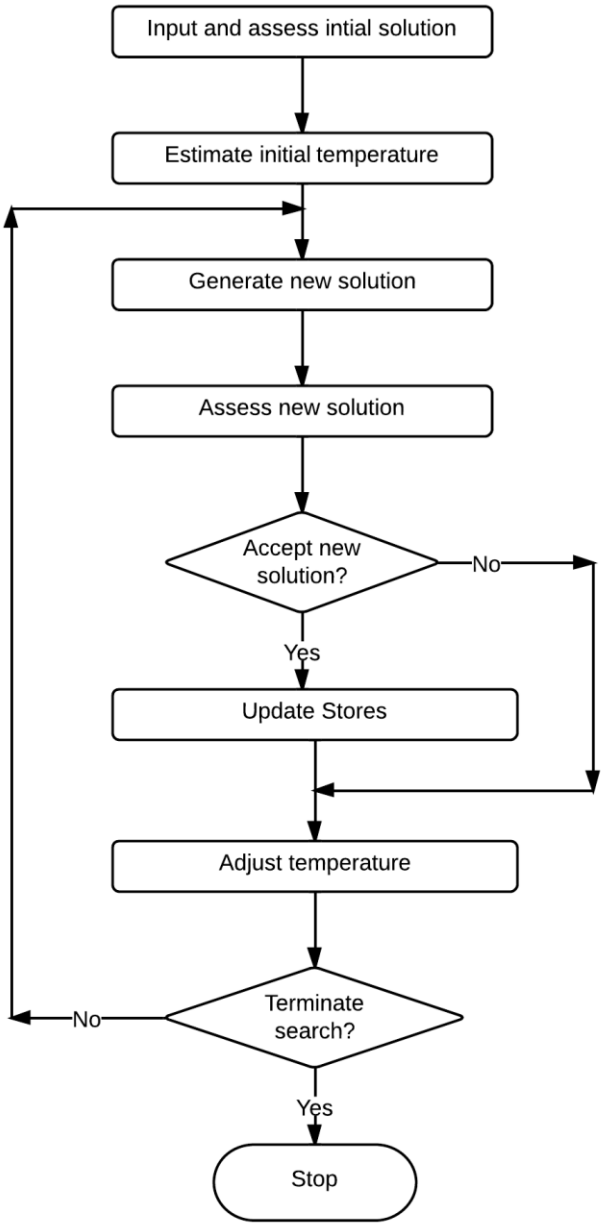

Figure 4: Flow chart illustrating working of SA

- SA methods are ‘tuneable’. Strategic oscillation and reheats can be used for to enhance its performance if more computational power is at disposal.

Weakness

- Since it’s a metaheuristic, lots of modification needs to be done to adapt it to a particular problem

- There exists a tradeoff between quality of solution and computational time required to find it.
- Fine tuning the parameter require empirical knowledge which comes with experience. Thus it require experience with SA for best results with a problem.
- The precision of parameter values used in SA can have significant impact on the final solution.

4.2.4 Comparison with other methods

Any efficient optimization algorithm must be good at two techniques to find a global maximum: investigate new and previously undiscovered areas in the search space, make use of knowledge gained during the search process till current time to explore better directions in search space. A tradeoff exists between these two techniques and a better metaheuristic must make better tradeoff choices than other.

Neural nets (NN)

- Learning is mandatory in neural networks but not in SA.
- SA is a flexible random search method while NN flexible function approximators.
- NN are adaptive in nature which makes the suitable for dynamic environments. SA is power hungry algorithm so cannot be used in real time applications.

Genetic Algorithms

- GA does not give any statistical gurantee to global convergence while it has been proved SA can find global optima if infinite time is given to it.

- Adaptive simulated annealing (ASA) has outperformed GA on a test suit adopted for both of them for a common problem.

Gradient methods

- These methods are applicable to only continuous because derivative needs to be computed to find gradient.
- These methods are often called as hill climbing methods. They are good for single peak functions but for many peak functions, SA outperform them.

Iterated Search

- It is the combination of gradient method with random search. The combination can be termed as iterated hill climbing because in each iteration a random point is chosen as starting point to discover the hill.
- It is good if function contains few hills.
- It does not give the overall picture of domain space searched while SA searches in domain in very systematic and connected manner.

Chapter 5
Detailed System Architecture

This chapter contains the design details of the system that we developed for finding minimum cost mTTP schedules. Designing of a system is critical part before a retractable solution can be obtained. Thus we focused on developing our system in modules which are independent of each other. The architecture is explained in fairly detailed way in this chapter. This would help the reader to understand our proposed approach given in next chapter.

We will first discuss the basic system overview. Then we will delve in to the detailed system design. After than we will explain the challenges and solution of each smallest module which is part of our project.

5.1 System Overview

Out system can be broadly understood by diagram given in Figure 5. It is a three step process. In first step, a *population* of feasible solutions is created. In second step, BBO is applied on the population to improve them and make them converge to a local optima quickly. In final step, simulated annealing is used to find near optimal solution using best solution of BBO. Below we describe the purpose of inputs, how they are utilized and which modules make use of them.

5.1.1 Inputs

We need following two inputs

1. Team size: We need to know the number of teams playing in tournament for two reasons

a. It lets us create the permutation of integers which will be used as input for polygon method to generate abstract schedules
b. It lets us choose the appropriate distance matrix for our problem

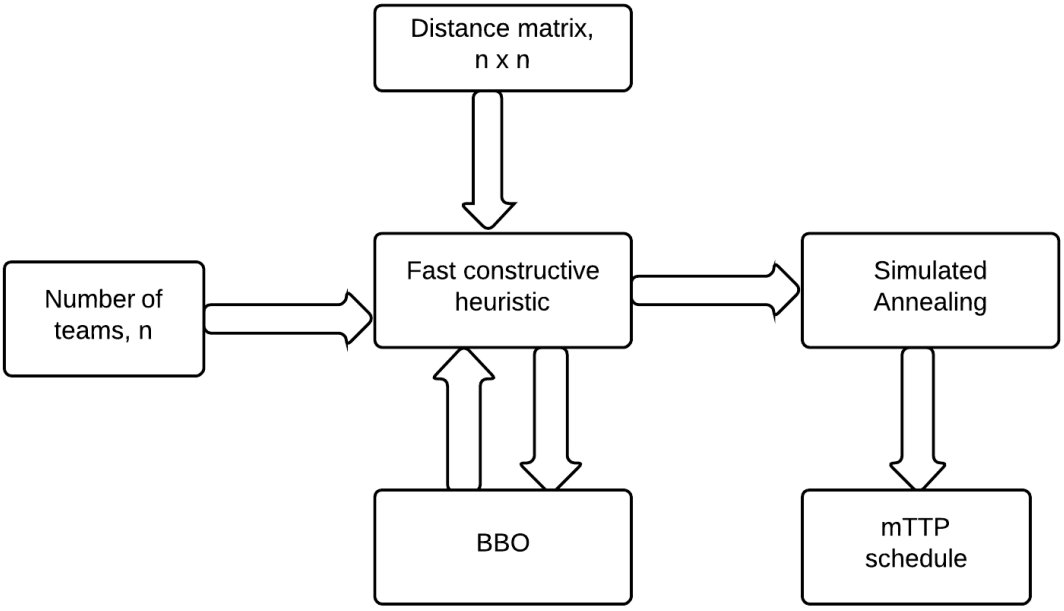

Figure 5: Block diagram of proposed approach

2. Distance matrix: This input also has two purposes
    a. It is used to calculate the cost of a schedule. In our work, cost is the only metric on which our schedules are compared.
    b. It is used by fast constructive heuristic to covert abstract schedules in to real schedules

5.1.2 Fast Constructive Heuristic improved with BBO

The idea of fast constructive heuristic is adapted from [8]. This uses polygon method to construct abstract (can also be though as random) schedules.

Most of work on mTTP before 2007 used only abstract schedules as input. But it had the drawback that distances among the team cities was completely ignored. Thus the

random schedules do not provide good starting point for optimizing metaheuristic which can have significant effect on its performance. Urrutia [8] introduced the concept of using distance for improving initial schedules and got significant improvement in cost minimization using just this single fact. He did not try to apply any further optimization on initial schedule generation. We carried this task further and applied BBO on Fast Constructive method of [8].

Because of the design of polygon method, out of infinitely many schedules possible for a give input, only limited number of schedules can be created. The flaw lies in rigid design of clockwise rotation. Thus number of unique schedules for a team size of *n* that can be created by using polygon method is (*n*-1)!.This is actually a very small number compared to all the possible schedules *i*.e. $2^{n^{2n-2}}$. As it seems, it is impossible to explore this much large search space so we tried to find the optimal solutions among the (*n*-1)! schedules which can be created by polygon method using BBO. BBO worked on the permutations used as input to polygon method which can be at most (*n*-1)! for an integer vector of size *n*.

5.1.3 Simulated Annealing

Simulated annealing has shown its mettle for generating near optimal solution for mTTP. TTP has double round robin (DRR) constraint. It is not easy for a population based heuristic like (GA, BBO, PSO etc.) to directly modify mTTP schedules as it can make them invalid. Once a schedule becomes invalid, it is impossible to make it valid in further iterations. DRR constraint is very rigid and difficult to follow. But neighborhood based algorithms like SA, tabu search, iterated hill climbing can explore the search space of mTTP schedules without breaking DRR constraint using predefined neighborhood moves. Thus even if a schedule becomes infeasible (breaks

At-most or No-repeat constraint), it can be resorted to a feasible one. But DRR structure is preserved by all the moves of SA. So we chose SA as final refinement algorithm. We just pass the best solution of BBO to SA to get the near optimal solution.

5.2 Detailed System Architecture

Figure 6 describes each and every small module of our proposed approach. Each module is moderately complex in itself thus we required to create multiple functions to implement one module. What follows is the description of these modules

5.2.1 Unique permutation generator

It is actually no so trivial to generate unique and random permutations from 1 to *n*, for any given value of *n*. We used the idea of Ged Ridgway [33]. Although there are *n*! permutations possible, we just use the 100 out of them. It is because, we use 100 members in our population for BBO algorithm.

5.2.2 Selection of appropriate distance matrix

Based on the value of input variable *n*, we append it with string 'NL' to create a new string 'NL*n*'. For example is input team size is 10, this module makes a string 'NL10'. We have stored the NL*n* instances data in text files. This module selects the appropriate text file based on above formed name.

5.2.3 Abstract schedule generation with polygon method

We follow the approach described in detail in section 6.1.1. The output of this module is a plain double round robin schedule. We have not assigned stadiums (signs to integers of matrix) yet. We try to make feasible and valid schedule in this step.

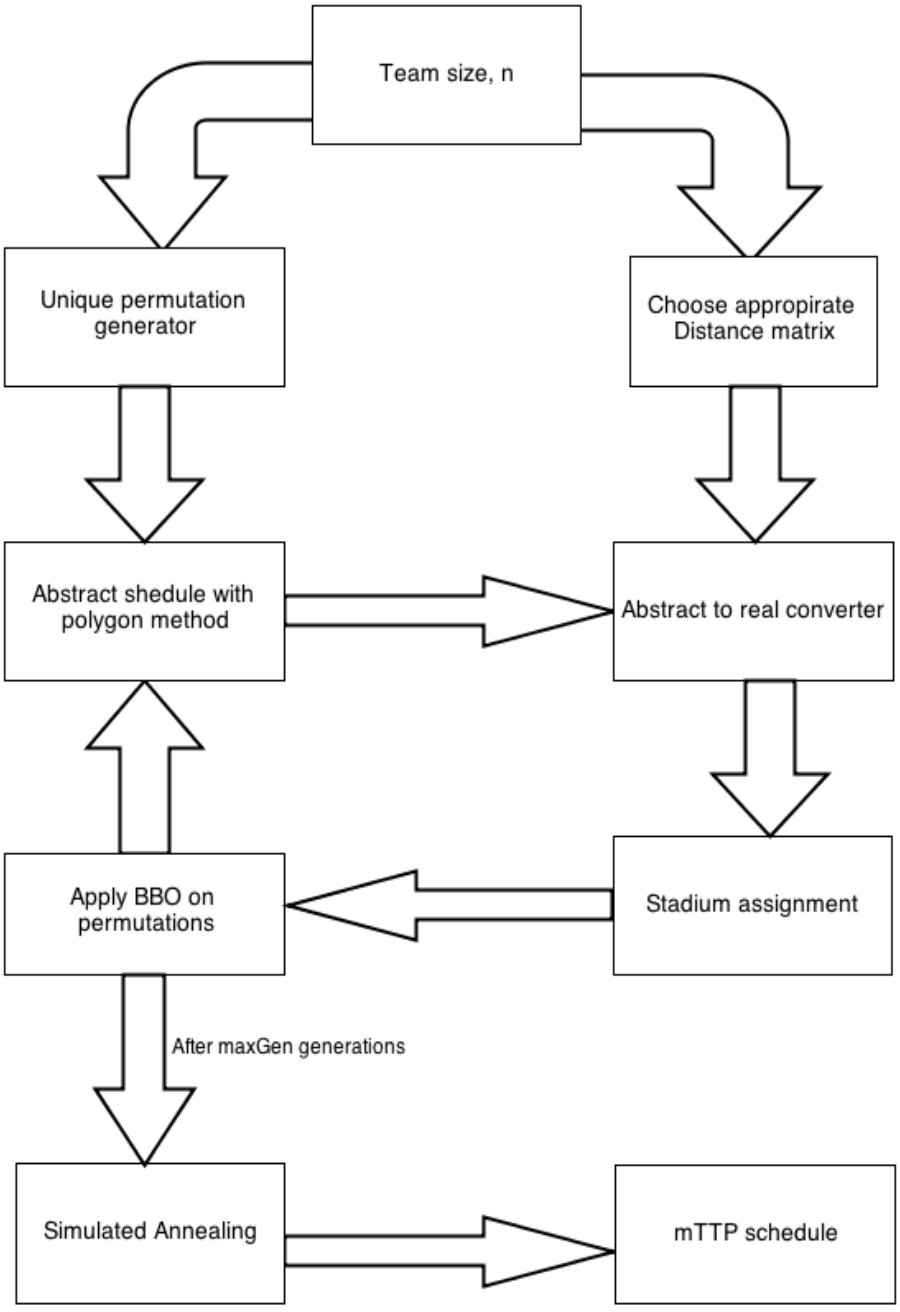

Figure 6: Architecture of our proposed approach

.

5.2.4 Abstract to real schedule converter

We need to do mapping to for this conversion to take place. If most of the team plays with two particular teams of the league in consecutive rounds, then it is rational to

assign closest distance real teams to those abstract teams as it would save travelling distance of many teams of tournament. Major steps involved in this module are:

1. Make a list of pair-of-teams which appear most frequently in abstract schedule.
2. Sort above list in ascending order
3. Make a list of size *n*, where each index of each element represents a id real team and value represents id of its closest team.
4. Assign list 2 teams to list 1 teams according to algorithm given in section 6.1.3

5.2.5 Stadium Assignment

We try to keep the road trip length (consecutive away matches) as long as possible because the previous step ensured that consecutively appearing teams in schedule have least possible distance. Thus in optimal case, we try to assign negative signs to three consecutive integers in a row, then one positive sign, then again three negative signs and so on. To ensure that we do not break the mTTP constraints while assigning stadiums, we follow the algorithm given in section 5.2.5

5.2.6 Apply BBO on permutations

Pervious step gives us a full-fledged and feasible mTTP schedule. With step, we begin our improvisation. We believe the abstract schedule is as good as the permutation from which it was formed. So we apply metaheuristic BBO on permutations to improve them which in return abstract schedules. This improvement is seen when we get abstract schedules whose consecutive opponent matrix has non uniform values throughout. A better real team assignment is possible for this kind of abstract schedule.

5.2.7 Simulated annealing

We apply standard simulated annealing with modified parameters (decided empirically) to the best solution of previous step. This helps us reaching to near optimal solution in less time as we give SA a better direction in the very first stage. The results confirmed the fact (which is yet to be proved mathematically) that a good starting point indeed makes the searching process of simulated annealing better and faster

The breakdown of the proposed system is presented in this chapter. All the inputs and outputs of each small and independent part of the system are discussed. The purpose of this chapter was to acquaintance the reader with the system as a black box i.e. what the system does, not how it does. The next chapter contains algorithms in details illustrating how the system actually works using the metaheuristics of our choice. In that, we will discuss our proposed approach and describe how the metaheuristics are adapted to suit our problem needs.

.

Chapter 6
Proposed Approach

In this chapter, we shall present a BBO based model for finding near optimal solution for mTTP. For doing this, we have adapted BBO to suit our application.

In chapter 5, we gave overview of the detailed architecture of our proposed system. Purpose of each module was described. In this chapter, we will give the inner working details of each module.

A good initial point for simulated annealing can improve its performance dramatically [8]. Thus we strive to have as good starting point as we can combining fast constructive heuristic and BBO. The three step process of our proposed approach is described with their algorithms. Following sections define each of those steps in detail.

6.1 Fast constructive heuristic for good initial solutions.

This process has three steps. First we create single round robin (SRR) abstract schedules using "polygon method" [34]. We can get the double round robin schedule by simple appending the above found SRR schedule with same SRR schedule having inverted signs. The abstract schedules created in first step are improved by assigning real teams to abstract teams based on distance matrix and frequency of consecutive matches played in abstract schedule. Stadiums are then assigned in step three by assigning signs to integers in mTTP schedule matrix. In each step, emphasis is done to minimize the cost. Following section describes above mentioned steps in detail.

6.1.1 Abstract Schedule Creation

To make an abstract schedule, we first generate an initial integer vector of *n* random and unique positive integers from 1 to *n*, example of which is shown in Figure 10.

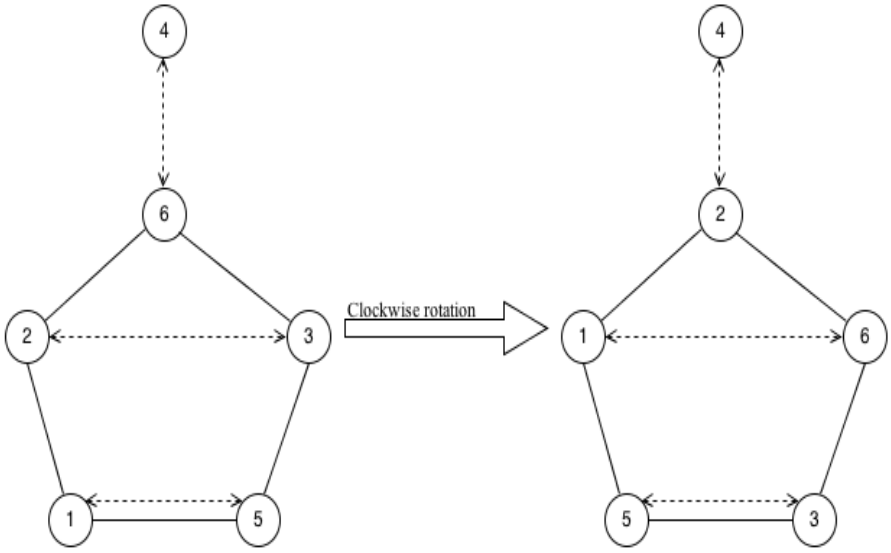

Figure 7: Rotation in polygon method for *n*=6

T\R	**1**	**2**	**3**	**4**	**5**
1	5	6	4	3	2
2	3	4	5	6	1
3	2	5	6	1	4
4	6	2	1	5	3
5	1	3	2	4	6
6	4	1	3	2	5

Figure 8: SRR schedule generated by polygon method

T\R	1	2	3	4	5	6	7	8	9	10
1	5	6	4	3	2	5	6	4	3	2
2	3	4	5	6	1	3	4	5	6	1
3	2	5	6	1	4	2	5	6	1	4
4	6	2	1	5	3	6	2	1	5	3
5	1	3	2	4	6	1	3	2	4	6
6	4	1	3	2	5	4	1	3	2	5

Figure 9: Abstract DRR obtained by appending SRR with itself

As we need to use BBO to refine schedules generated in this step, we have to make a population of permutations. Thus we generate *P* number of random permutations initially, where *P* defines the number of population members to be used in BBO. Each permutation must be unique. The permutation from which SRR of Figure 8 is obtained is given in Figure 10 below.

6	3	5	1	2	4

Figure 10: Example of permutation used for polygon method

We place integers from this permutation on the *n*-1 nodes of a regular polygon consecutively as shown in Figure 7. The nth integer is placed outside of polygon. In each round *k* from 1 to *n* -1, team at node l = 2,4....*n*/2 plays with the team at node $n+l-1$. This way, each team plays with a team located symmetrically opposite to it in the polygon. After a round is completed, each of the *n*-1 teams are rotated clockwise to take immediate next location in polygon. Thus in *n*-1 rounds, we get a SRR schedule shown in Figure 8. We duplicate this schedule to get a mirrored DRR illustrated by Figure 9. Now we need to convert this abstract schedule to a schedule whose entries reflects real team numbers.

6.1.2 Abstract to real team assignment

The mirrored DRR schedule generated in step 1 does not consider distances among teams. We strive to turn these abstract schedules into good quality schedules by assigning those pair of abstract teams to real teams who play consecutive matches maximum times in abstract schedule.

$$\begin{pmatrix}
0 & 4 & 4 & 4 & 4 & 4 & 4 & 3 & 4 & 4 & 3 & 4 & 4 & 4 & 4 & 4 \\
4 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 \\
4 & 2 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 \\
4 & 25 & 2 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 25 & 2 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 25 & 2 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 25 & 2 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 2 & 26 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 1 & 26 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 26 & 1 & 0 & 2 & 25 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 26 & 2 & 0 & 2 & 25 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 2 & 25 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 2 & 25 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 2 & 25 \\
4 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 2 \\
4 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0
\end{pmatrix}$$

Figure 11: Consecutive opponents matrix for n=16

Thus the overall travelling cost of tournament will decrease as maximum consecutive matches will be played between teams having smaller distance between their venues. To make this possible, we first need to generate a consecutive opponent matrix. Value (i,j) of opponent matrix (example of which is shown in Figure 11) tells the number of times abstract team T_i and T_j play against each other consecutively during the whole tournament. For example, during whole tournament, some team plays with team 4 and team 2 in consecutive rounds 25 times.We keep a list $L1$ of size $n(n\text{-}1)/2$ containing values in upper half triangle of opponent matrix in ascending order. A list $L2$ containing $n(n\text{-}1)/2$ unique pair of teams is used in which

each pair has a distance associated with it which represents home distance between pair. This list is sorted in ascending order. Then according to following algorithm, we assign abstract to the real team [8].

Let t_1 be the next real team which is to be assigned to an abstract team and let t_2 is the team having closest distance from t_1:

- If t_2 is already assigned to an abstract team, then find the first pair (a, b) of abstract teams such that a is assigned to t_2 and b is not yet assigned to a real team. In this case, assign the abstract team b to the real team t_1.
- If t_2 is not yet assigned to an abstract team, then find the first pair (a, b) of abstract teams such that none of them is assigned to a real team. In this situation, assign either the abstract team a to the real team t_1 and the abstract team b to the real team t_2 or vice versa.

6.1.3 Stadium Assignment

In this round we assign the stadiums to matches *i*.e. from the two participating teams, at which team's venue match will be played. It basically means assigning signs to integers in scheduled generated so far. Step 2 tried to minimize the travel distance for a team if all its matches were played on road trip. A road trip means consecutive number of matches that are played away. Home stand is vice versa. Thus we try to schedule maximum matches on a road trip to keep the travel cost lower without breaking the At-most feasibility constraint. We use following strategy to assign stadiums to our schedule [8].

Stadiums are randomly assigned to the games of first round. As first round contain *n* teams, *n*/2 negative signs are assigned randomly to *n* integers of first column. From round 2 to *n*-2, we use the following strategy to assign a stadium to

the game between t_1 and t_2. We denote by n_i the number of games that team *i* has consecutively played in the previous rounds, either in a home stand or in a road trip.

Case *I*: $n_{t2} > n_{t1}$

- if team t_2 played its last game at home, then schedule the game to the stadium of team t_1;
- Otherwise, schedule the game to the stadium of team t_2.

Case II: $n_{t2} < n_{t1}$

- if team t1 played its last game at home, then schedule the game to the stadium of team t2;
- Otherwise, schedule the game to the stadium of team t1.

Case III: $n_{t2} < nt1$

- if team t1 played its last game at home and team t2 away, then schedule the game to the stadium of team t2;
- if team t2 played its last game at home and team t1 away, then schedule the game to the stadium of team t1;
- Otherwise, schedule the game randomly to the stadiums of t1 or t2.

6.2 Adapted BBO for fast convergence to local optima

We apply BBO on the permutations that are used to generate abstract schedules. A good permutation is able to generate an abstract schedule which when converted to a real will have lesser travel cost. We treat each integer in the permutation as an *SIV* and travel cost of schedule generated from this permutation as HSI. As a first step

of applying BBO, we sort the habitats from best to worst based on their HSIs. Then we get the species count by using the Algorithm 1 given in Fig 6.

Based on species count, immigration rate λ and emigration rate μ are calculated using (3) and (4).

$$\mu_i = \frac{E}{N}\,(E * k) \tag{3}$$

$$\lambda_i = I * \left(1 - \frac{k}{N}\right) \tag{4}$$

Here *E* is maximum emigration rate, *I* is maximum immigration rate, *k* is number of species on ith habitat and *N* is total number of species in the population [11]. In our model, *I* and E are chosen to be 1.

For each habitat, its *SIV* is modified with probability λ. Let's say based on λ, only one *SIV* is selected to be modified in i^{th} habitat. Its position in i^{th} habitat is *k*. Then with probability μ we choose a habitat which will give its k^{th} *SIV* to the i^{th} habitat. The emigrating habitat then copies its k^{th} *SIV* at the k^{th} position in i^{th} habitat.

SIVs	***SIV1***	***SIV2***	***SIV3***	***SIV4***	***SIV5***	***SIV6***
Island H1	2	1	4	3	6	5
Island H2	4	2	6	5	1	3

a) Before migration

Island H1	2	1	4	3	6	5
Island H2	4	2	1	5	6	3

b) After migration

Figure 12: Migration operation in BBO-TTP

Rest of the *SIV*s of i^{th} habitat are adjusted to meet the feasibility requirement of permutation *i*.e. all integers must be unique and from 1 to *n* only. A good habitat tends to be more resistant to change than a bad habitat. Figure 12 shows how migration process occur in BBO for TTP

The explanation of migration operation depicted above (in Figure 12) is given below:

1. Using λ of habitat 2, its *SIV*5 is chosen to be modified.
2. Now using roulette wheel selection method on μ of all habitats, we try to find the habitat which will emigrate its *SIV*5 to habitat 2. Here, it comes out to be habitat 1.
3. So habitat 1 replaces *SIV*5 of habitat 2 with value 6. As two integers with same value cannot exist in our permutation, *SIV*3 of habitat 2 is replaced with older *SIV*5 of habitat 2 *i*.e. value 1. Habitat 1 is not modified in this process. BBO for TTP is given in Figure 14.

6.2.1 Get-Species-Count Algorithm

The algorithm used for mapping cost of a solution to its species count is at the heart of BBO. It is important because species count is used to find out HSI, immigration rate and emigration rate. BBO relies on these three values for migration process which helps in improving solution.

Algorithm 1: GetSpeciesCount(P)

```
1. totalCost = Calculate total cost by adding costs of all schedules in population.
2. currentCost = totalCost
3. P(1).speciesCount = currentCost
4. for i = 2 to sizeof(P)
5.     currentCost = currentCost – P(i).cost
6.     P(i).speciesCount = currentCost
7. end for
```

Figure 13: Algorithm to map cost to species count in BBO

Below we explain the get species count algorithm adapted for mirrored traveling tournament problem. This algorithm assumes that population is already sorted from best to worst.

Explanation of GetSpeciesCount Algorithm given in Figure 13

1. Find the total cost by summing traveling costs of all the schedules in population.
2. We find the species count of a member by subtracting cumulative cost of members better than it from the total cost calculated in step one. For this to implement, we keep a variable current cost and initialize it with total cost. This cost keeps on decreasing when we iterate through the population from best to worst.
3. Give the total cost value as species count to 1st (and the best one) member of population.
4. Iterate from 2nd member till end of members list.
5. Modify current cost by subtracting current schedule cost from variable CurrentCost.
6. Assign current cost value as species count to member in consideration
7. Go to step 4. Repeat this until all members are explored.

6.2.2 BBO Algorithm adapted for TTP

In Figure 14, the adapted BBO for traveling tournament problem is illustrated. As in our proposed approach, BBO and fast constructive heuristic (FCH) works in collaboration, schedule generation, wherever needed, is done by FCH in this algorithm. All data structures with their purpose are described in comments and each step of the algorithm is explained point wise below Figure 14.

Algorithm 2: BBO-TTP

1. Generate the random population, *P*, of habitats
2. Calculate HSI of each habitat in *P*
3. **for** *i* = 1 to MaxGen
4. Sort the population from best to worst
5. For each habitat, map HSI to species count.
6. Calculate immigration rate λ and emigration rate μ for each habitat
7. ReplaceableIndices = [] % Stack containing indices of current habitat to % be replaced by emigrating habitat
8. *PreviousHabitat* = *P(i)* %save habitat before modifying it
9. **for** *j* = 1 to length of Population member
10. If i^{th} *SIV* of current habitat, *CH*, is ready for immigration according to member's λ
11. Push *j* to ReplaceableIndices
12. **end for**
13. Choose the emigrating habitat, *EH*, from population using μ
14. **while** stack ReplaceableIndices is not empty
15. *j* = pop(ReplaceableIndices)
16. Migrate j^{th} *SIV* of *EH* to *CH*
17. **end while**
18. Try to get new schedule using modified *CH*
19. **If** it is not feasible to generate schedule using modified *CH* or if cost of *CH* schedule is more than *PreviousHabitat*'s cost
20. Replace newly formed habitat with *PreviousHabitat*
21. **end if**
22. **end for**

Figure 14: BBO-TTP algorithm

Abstract working of BBO in solving TTP is shown in Figure 15. Below we give step wise explanation of BBO-TTP algorithm

1. Use fast constructive heuristic (FCH) proposed by [8] to develop initial schedules. We have to call FCH *p* times for *p* random permutations where *p* is the number of members in population that we intend to have.

2. Calculate the cost of each habitat. It corresponds to the traveling cost of schedules generated in step 1.

3. Iterate steps 3rd to 22nd MaxGen times where MaxGen is the number of iterations (generations) for which we will run BBO

4. Sort the population of schedules based on their traveling cost

5. Call GetSpeciesCount algorithm for each of the current population member (current habitat) CH to map cost to species count.

6. Calculate immigration rate and emigration rate of each member based on species count. We use equations (3) and (4) for it.

7. Keep a stack named ReplacableIndices that would contain the indices of SIVs chosen to be replaced my immigration from current habitat.

8. Save data of current population member before modifying it.

9. Scan all SIV's of current population member

10. If according to its λ, current SIV is ready to be replaced then

11. Push its index in to the ReplacableIndices stack

12. Go to step 10 until all SIV's of current population are scanned.

13. Using roulette wheel selection method based on their μ choose the emigrating habitat (EH) among the population which will give its SIV's to current population member.

14. Iterate through the stack

15. Pop value at top of the stack in to a variable '*j*'

16. Migrate j^{th} SIV of EH to CH

17. Go to step 14 until stack gets empty

18. Try to generate new schedule using FCH from permutation modified by BBO

19. If it is not feasible to generate schedule using modified permutation or if cost of new schedule if more than PreviousHabitat's schedule, then

20. .Replace modified member with PreviousHabitat

21. Close step 19th if logic

22. Go to step 3rd.

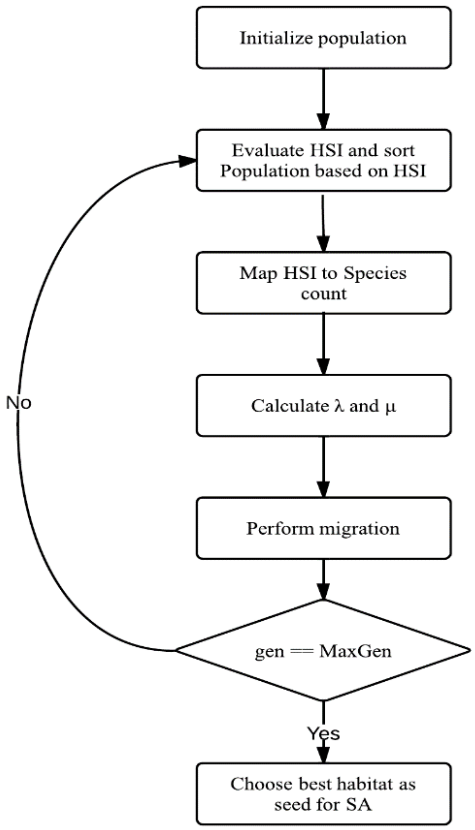

Figure 15: Flow chart showing working of BBO-TTP

BBO works on the permutations used to generate abstract schedules. It tries to improve abstract schedules by migrating SIVs of good habitats to habitats having low HSI. This process is carried out for each habitat of population in each generation until the generation count reaches to MaxGen. After each generation, some habitats

improve so their HSI are recalculated before applying BBO in next generation. Algorithm given in Figure 13 is used to get species count from cost of the schedule.

6.3 Using Simulated Annealing to refine solution

Simulated annealing is a local search heuristic which has been applied on TTP earlier and achieved good results [17], [22].

We use following three simple neighborhood structures on the best solution of BBO. A neighborhood of a TTP schedule S is a schedule which can be obtained by applying any of the three types of simple moves defined by Anagnostopoulos [17]:

1. *SwapHomes(S, T_i, T_j)*: Say if team T_i plays home with team T_j in round *k*, and at T_j's home in round *l*, then by applying this move, we will get a schedule in which T_i plays home with T_j in round *l* and at *Tj*'s home in round *k*. For this move, we need to modify four values in *S*. Example is shown in Figure 16.

	First Half					**Second Half**				
T\R	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***
1	-5	-6	3	-2	4	5	6	-3	2	-4
2	3	5	-4	1	6	-3	-5	4	-1	-6
3	-2	4	-1	6	5	2	-4	1	-6	-5
4	6	-3	2	5	-1	-6	3	-2	-5	1
5	1	-2	6	-4	-3	-1	2	-6	4	3
6	-4	1	-5	-3	-2	4	-1	5	3	2

	First Half					Second Half				
T\R	1	2	3	4	5	6	7	8	9	10
1	-5	-6	3	-2	4	5	6	-3	2	-4
2	3	-5	-4	1	6	-3	5	4	-1	-6
3	-2	4	-1	6	5	2	-4	1	-6	-5
4	6	-3	2	5	-1	-6	3	-2	-5	1
5	1	2	6	-4	-3	-1	-2	-6	4	3
6	-4	1	-5	-3	-2	4	-1	5	3	2

Figure 16: *SwapHomes(S,2,5).* Before move (upper), After move(lower)

2. *SwapRounds(S, r_k, r_l)*: This move simply swap teams of rounds r_k with teams of round r_l in *S*. It modifies 2**n* values in *S*. Example is shown in Figure 17

	First Half					Second Half				
T\R	1	2	3	4	5	6	7	8	9	10
1	-5	-6	3	-2	4	5	6	-3	2	-4
2	3	5	-4	1	6	-3	-5	4	-1	-6
3	-2	4	-1	6	5	2	-4	1	-6	-5
4	6	-3	2	5	-1	-6	3	-2	-5	1
5	1	-2	6	-4	-3	-1	2	-6	4	3
6	-4	1	-5	-3	-2	4	-1	5	3	2

	First Half					Second Half				
T\R	1	2	3	4	5	6	7	8	9	10
1	-5	-6	4	-2	3	5	6	-4	2	-3
2	3	-5	6	1	-4	-3	5	-6	-1	4
3	-2	4	5	6	-1	2	-4	-5	-6	1
4	6	-3	-1	5	2	-6	3	1	-5	-2
5	1	2	-3	-4	6	-1	-2	3	4	-6
6	-4	1	-2	-3	-5	4	-1	2	3	5

Figure 17: *SwapRounds(S, 3, 5).* Before move (upper), after move (lower)

3. *SwapTeams(S, Ti, Tj)*: The schedules of Teams T_i and T_j are swapped using this move expect for two rounds in which they play with each other. This move modifies 2*(2*n*-4) values in *S.* Example is shown in Figure 18.

Three moves given above always produce a valid double round schedule. Although At-most and No-repeat constraints may get violated by these moves. Schedules produces by violation of these constraints are called infeasible schedules although they are still valid DRR schedules. SA balances time between exploration of infeasible and feasible regions using reheats and strategic oscillations.

	First Half					Second Half				
T\R	1	2	3	4	5	6	7	8	9	10
1	-5	-6	3	-2	4	5	6	-3	2	-4
2	3	5	-4	1	6	-3	-5	4	-1	-6
3	-2	4	-1	6	5	2	-4	1	-6	-5
4	6	-3	2	5	-1	-6	3	-2	-5	1
5	1	-2	6	-4	-3	-1	2	-6	4	3
6	-4	1	-5	-3	-2	4	-1	5	3	2

	First Half					Second Half				
T\R	1	2	3	4	5	6	7	8	9	10
T1	6	-3	2	5	4	-6	3	-2	-5	-4
T2	3	5	-1	4	6	-3	-5	1	-4	-6
T3	-2	1	-4	6	5	2	-1	4	-6	-5
T4	-5	-6	3	-2	-1	5	6	-3	2	1
T5	4	-2	6	-1	-3	-4	2	-6	1	3
T6	-1	4	-5	-3	-2	1	-4	5	3	2

Figure 18: SwapTeams(S, 1, 4). Before move (upper), after move (lower)

Number of violations play a role in determining cost of schedule *S*. By considering violations in cost function, time spent in infeasible and feasible region can be balanced.

$$C(S) = \begin{cases} cost(S)\ if\ S\ is\ feasible, \\ \sqrt{cost(S)^2 + \ [w.f\big(nbv(S)\big)]^2}\ otherwise \end{cases} \quad (5)$$

Here *cost(S)* is the travel cost of schedule, *w* is weight parameter *nbv(S)* are number of At-most and No-repeat violation is *S* and *f* is a sub-linear function which is chosen to be:

$$f(x) = \ 1 + \left(\sqrt{x}\ \ln x\right) \quad (6)$$

Local optima which might occur at very low temperature is avoided using reheats and strategic oscillations. Simulated annealing algorithm for TTP is given in Figure 19.

```
1.  Get best schedule S of BBO;
2.  bestFeasible ← ∞; nbf ← ∞;
3.  bestInfeasible ← ∞; nbi ← ∞;
4.  reheat ← 0; counter ← 0;
5.  while reheat ≤ maxR do
6.    phase ← 0;
7.    while phase ≤ maxP do
8.      counter ← 0;
9.      while counter ≤ maxC do
10.       select a random move m from neighborhood(S);
11.       let S' be the schedule obtained from S with m;
12.       if C(S') < C(S) or
13.         nbv(S') == 0 and C(S') < bestFeasible or
14.         nbv(S') > 0 and C(S') < bestInfeasible
15.       then
16.         accept ← true;
17.       else
18.         accept ← true with probability exp(−ΔC/T),
19.                  false otherwise;
20.       end if
21.       if accept then
22.         S ← S';
23.         if nbv(S) == 0 then
24.           nbf ← min(C(S), bestFeasible);
25.         else
26.           nbi ← min(C(S), bestInfeasible);
27.         end if
28.         if nbf < bestFeasible or nbi < bestInfeasible then
29.           reheat ← 0; counter ← 0; phase ← 0;
30.           bestTemperature ← T;
31.           bestFeasible ← nbf;
32.           bestInfeasible ← nbi;
33.           if nbv(S) == 0 then w ← w/θ; else w ← w · δ; end if
34.         else
35.           counter++;
36.         end if
37.       end while
38.     phase++;
39.     T ← T · β;
40.   end while
41.   reheat++;
42.   T ← 2 · bestTemperature;
43. end while
```

Figure 19: Simulated Annealing algorithm for mTTP

Simulated annealing procedure returns our solution, i.e. best mTTP schedule that our hybrid approach could find. The majority of time our approach take is consumed by SA. But as the starting point of SA was very good, we avoided a large part of not-so-useful search space from the huge search space that SA needs to explore. This helped in finding better solution in smaller time. We will now give the results our approach took on standard benchmarks in next chapter.

Chapter 7
Experiments and Results

This chapter contains the discussion of the environment setup used to find results for our mirrored traveling tournament problem. We describe the input data used by our algorithm, the data it produce as output. We will compare our results with the current best known heuristics and analyse our work.

The proposed hybrid algorithm is applied on National League (NL) instances and real world instance of Brazilian soccer league (BRA24) available on [10]. The NL*n* instances are based on Major League Baseball (MLB) are were derived by Easton *et al.* [1]. BRA24 instance is real life data of Brazilian soccer league gathered by Urrutia *et al.* [8]. These are the standard datasets for TTP. The best results of all time in chronological order is given on [10]. Anyone can challenge the best solutions with his/her by mailing his/her schedule to the author of TTP website [10].

7.1 Experimental setup used in this work

For each instance 50 generations of BBO each with 100 population members are used. We used Intel Pentium IV 3.0 Giga hertz machine with 1 GB of RAM. Although better machines are available today, we chose these parameters to set fair comparison basis for our results with results in literature. Simulated annealing parameters for solving TTP and are given in Table 2.

.

Table 2:
Simulated annealing parameters for mTTP

n	T_0	β	w_0	δ	θ	$maxC$	$maxP$	$maxR$	γ
8	400	0.9999	4000	1.04	1.04	5000	7100	10	2
10	400	0.9999	6000	1.04	1.04	5000	7100	10	2
12	600	0.9995	10000	1.03	1.03	4000	1385	50	1.6
14	600	0.9999	20000	1.03	1.03	4000	7100	30	1.8
16	700	0.9999	60000	1.05	1.05	10000	7100	50	2

7.2 Data set used for mTTP schedules generation

In this section we give the distance matrices of National League (NL*n*) instance which were used to generate mTTP schedules. These matrices represents air-to-air distance from the center of cities which are part of the Major League Baseball (MLB).

Table 3:
Distance matrix of 8 team instance of National League

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0	745	665	929	605	521	370	587
2	745	0	80	337	1090	315	567	712
3	665	80	0	380	1020	257	501	664
4	929	337	380	0	1380	408	622	646
5	605	1090	1020	1380	0	1010	957	1190
6	521	315	257	408	1010	0	253	410
7	370	567	501	622	957	253	0	250
8	587	712	664	646	1190	410	250	0

Table 4:
Distance matrix of 10 team instance of National League

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	745	665	929	605	521	370	587	467	670
2	745	0	80	337	1090	315	567	712	871	741
3	665	80	0	380	1020	257	501	664	808	697
4	929	337	380	0	1380	408	622	646	878	732
5	605	1090	1020	1380	0	1010	957	1190	1060	1270
6	521	315	257	408	1010	0	253	410	557	451
7	370	567	501	622	957	253	0	250	311	325
8	587	712	664	646	1190	410	250	0	260	86
9	467	871	808	878	1060	557	311	260	0	328
10	670	741	697	732	1270	451	325	86	328	0

Table 5:
Distance matrix of 12 team instance of National League

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	0	745	665	929	605	521	370	587	467	670	700	1210
2	745	0	80	337	1090	315	567	712	871	741	1420	1630
3	665	80	0	380	1020	257	501	664	808	697	1340	1570
4	929	337	380	0	1380	408	622	646	878	732	1520	1530
5	605	1090	1020	1380	0	1010	957	1190	1060	1270	966	1720
6	521	315	257	408	1010	0	253	410	557	451	1140	1320
7	370	567	501	622	957	253	0	250	311	325	897	1090
8	587	712	664	646	1190	410	250	0	260	86	939	916
9	467	871	808	878	1060	557	311	260	0	328	679	794
10	670	741	697	732	1270	451	325	86	328	0	1005	905
11	700	1420	1340	1520	966	1140	897	939	679	1005	0	878
12	1210	1630	1570	1530	1720	1320	1090	916	794	905	878	0

Table 6:
Distance matrix of 14 team instance of National League

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**12**	**14**
1	0	745	665	929	605	521	370	587	467	670	700	1210	2130	1890
2	745	0	80	337	1090	315	567	712	871	741	1420	1630	2560	2430
3	665	80	0	380	1020	257	501	664	808	697	1340	1570	2520	2370
4	929	337	380	0	1380	408	622	646	878	732	1520	1530	2430	2360
5	605	1090	1020	1380	0	1010	957	1190	1060	1270	966	1720	2590	2270
6	521	315	257	408	1010	0	253	410	557	451	1140	1320	2260	2110
7	370	567	501	622	957	253	0	250	311	325	897	1090	2040	1870
8	587	712	664	646	1190	410	250	0	260	86	939	916	1850	1730
9	467	871	808	878	1060	557	311	260	0	328	679	794	1740	1560
10	670	741	697	732	1270	451	325	86	328	0	1005	905	1846	1731
11	700	1420	1340	1520	966	1140	897	939	679	1005	0	878	1640	1300
12	1210	1630	1570	1530	1720	1320	1090	916	794	905	878	0	947	832
13	2130	2560	2520	2430	2590	2260	2040	1850	1740	1846	1640	947	0	458
14	1890	2430	2370	2360	2270	2110	1870	1730	1560	1731	1300	832	458	0

7.3 Results

Comparison of results obtained by our hybrid approach are given in Table 10.

The values in columns (2 to 4) represent the traveling cost of best mTTP schedule obtained. Results are quite encouraging because it took significantly less time to reach to almost equal cost solutions from other methods. Our results have been verified from validator available official TTP website. None of the generated mTTP schedules show violation of any of the TTP or mTTP constrains.

Table 7:
Distance matrix of 16 team instance of National League

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	745	665	929	605	521	370	587	467	670	700	1210	2130	1890	1930	1592
2	745	0	80	337	1090	315	567	712	871	741	1420	1630	2560	2430	2440	2144
3	665	80	0	380	1020	257	501	664	808	697	1340	1570	2520	2370	2390	2082
4	929	337	380	0	1380	408	622	646	878	732	1520	1530	2430	2360	2360	2194
5	605	1090	1020	1380	0	1010	957	1190	1060	1270	966	1720	2590	2270	2330	1982
6	521	315	257	408	1010	0	253	410	557	451	1140	1320	2260	2110	2130	1829
7	370	567	501	622	957	253	0	250	311	325	897	1090	2040	1870	1890	1580
8	587	712	664	646	1190	410	250	0	260	86	939	916	1850	1730	1740	1453
9	467	871	808	878	1060	557	311	260	0	328	679	794	1740	1560	1590	1272
10	670	741	697	732	1270	451	325	86	328	0	1005	905	1846	1731	1784	1458
11	700	1420	1340	1520	966	1140	897	939	679	1005	0	878	1640	1300	1370	1016
12	1210	1630	1570	1530	1720	1320	1090	916	794	905	878	0	947	832	830	586
13	2130	2560	2520	2430	2590	2260	2040	1850	1740	1846	1640	947	0	458	347	654
14	1890	2430	2370	2360	2270	2110	1870	1730	1560	1731	1300	832	458	0	112	299
15	1930	2440	2390	2360	2330	2130	1890	1740	1590	1784	1370	830	347	112	0	358
16	1592	2144	2082	2194	1982	1829	1580	1453	1272	1458	1016	586	654	299	358	0

The mTTP schedules for NL12 and above instances have more than 22 columns and are difficult to present in the limited space available here. Below we compare our results with the best results and the results of popular metaheuristic Genetic Algorithm hybridized with simulated annealing. Results are quite competitive with best results in literature with cost margin barely crossing 5 percent. The result in bold represents the second best results up to the knowledge in literature [10]. Column T (sec) represents the time taken by BBO-SA to produce results. Mostly the current

best results are obtained in [22]. The maximum gap in schedule's cost between theirs and ours approach is just 5.4 %. The longest time our approach took was 10508 seconds for largest NL*n* instance while in [22] longest computation time reached 70898.90 seconds. Our hybrid approach produced competitive results in much lesser time. Fast convergence of solution using BBO helped in reaching to better solutions using simulate annealing in less time.

Table 8:
mTTP schedule for NL8

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	8	7	-5	3	2	4	-6	-8	-7	5	-3	-2	-4	6
2	4	8	6	-7	-1	-5	3	-4	-8	-6	7	1	5	-3
3	-6	4	-7	-1	-5	8	-2	6	-4	7	1	5	-8	2
4	-2	-3	8	6	-7	-1	-5	2	3	-8	-6	7	1	5
5	7	6	1	-8	3	2	4	-7	-6	-1	8	-3	-2	-4
6	3	-5	-2	-4	8	7	1	-3	5	2	4	-8	-7	-1
7	-5	-1	3	2	4	-6	8	5	1	-3	-2	-4	6	-8
8	-1	-2	-4	5	-6	-3	-7	1	2	4	-5	6	3	7

Cost of the schedule given above in Table 8 is 42802.

The mTTP schedules for NL12 and above instances have more than 22 columns and are difficult to present in the limited space available here. Below we compare our results with the best results and the results of popular metaheuristic Genetic Algorithm hybridized with simulated annealing. Results are quite competitive with best results in literature with cost margin barely crossing 5 percent. The result in bold represents the second best results up to the knowledge in literature [10]. Column T (sec) represents the time taken by BBO-SA to produce results. Mostly the current best results are obtained in [22]. The maximum gap in schedule's cost between theirs and ours approach is just 5.4 %. The longest time our approach took was 10508 seconds for largest NLn instance while in [22] longest computation time reached

70898.90 seconds. Our hybrid approach produced competitive results in much lesser time. Fast convergence of solution using BBO helped in reaching to better solutions using simulate annealing in less time.

Table 9:
mTTP schedules of NL10

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
1	-9	-8	10	4	2	-3	-7	-6	5	9	8	-10	-4	-2	3	7	6	-5
2	3	-4	-6	7	-1	-5	-9	8	10	-3	4	6	-7	1	5	9	-8	-10
3	-2	6	4	-5	7	1	-10	-9	8	2	-6	-4	5	-7	-1	10	9	-8
4	6	2	-3	-1	-5	7	8	10	-9	-6	-2	3	1	5	-7	-8	-10	9
5	-10	-9	-8	3	4	2	-6	-7	-1	10	9	8	-3	-4	-2	6	7	1
6	-4	-3	2	-8	-9	-10	5	1	-7	4	3	-2	8	9	10	-5	-1	7
7	8	10	9	-2	-3	-4	1	5	6	-8	-10	-9	2	3	4	-1	-5	-6
8	-7	1	5	6	-10	9	-4	-2	-3	7	-1	-5	-6	10	-9	4	2	3
9	1	5	-7	10	6	-8	2	3	4	-1	-5	7	-10	-6	8	-2	-3	-4
10	5	-7	-1	-9	8	6	3	-4	-2	-5	7	1	9	-8	-6	-3	4	2

Cost of schedule given above in Table 9 is 6663

Table 10: Comparison of Results

Instances	**Best Known**	**GA-SA**	**BBO-SA**	**T (sec)**	**Gap (%)**
NL8	41928	43112	**42805**	1435	2.09
NL10	63832	66264	66331	2425	3.76
NL12	119608	120981	121070	8665	1.20
NL14	199363	208086	210132	5609	5.40
NL16	278305	290188	291394	10503	4.70
BRA24	500756	511256	513543	42869	2.55

Chapter 8
Conclusion

In this work we tried to solve benchmark instances of mTTP while keeping the computation time low. Often in real world situations, we need a number of alternative schedules quickly so as to give lucrative time slots to popular teams. An approach taking time in number of days would not be feasible to use in these situations.

BBO takes very less time to reach to local optima. Although BBO is itself a global optimization technique, but because of its fundamental philosophy of sharing features among solutions, it is very difficult to apply it directly to mTTP schedules without breaking DRR constraint. Nevertheless, its application helped in improving the fast constructive heuristic schedules to a great extent. With just simple neighborhood moves of SA, we could get these competitive results. We could get the result of every instance tested which is not so easy for all metaheuristics. Our approach provided a strong alternative as a general sports scheduling technique. With more complex moves, a much larger neighborhood can be explored that might help in fine tuning the results.

Although our hybrid approach looks promising, but it did not performed well on large instances. When the search space is large, a huge penalty is paid for every wrong decision in searching. Our limited number of neighbourhood exploration moves restricted us to explore several good solutions. Also we did not consider mTTP as a soft constraint but a hard constraint in our approach. If we allow to break it during search process, we might reach at better solutions. A more complicated neighbourhood structure and a more liberal approach towards constrains is needed

to explore the best spots in search space. Although it sounds simple, to apply them with currently available metaheuristic and sport scheduling techniques is rather difficult.

To be used as a real life sports scheduling technique, more constraints can be added to our technique. We leave it for future work.

References

[1] K. Easton, G. Nemhauser and M. Trick, "The traveling tournament problem: Description and benchmarks," in *Principles and Practice of Constraint Programming, Lecture Noes in Computer Science*, vol. 2239, T. Walsh, Ed., Springer, 2001, pp. 580-584.

[2] F. D. Croce and D. Oliveri, "Scheduling the Italian football league: an ILP-based approach," *Computers and Operations Research,* vol. 33, no. 7, pp. 1963-1974, July 2006.

[3] M. Wright, "Scheduling fixtures for basketball New Zealand," *Computers & Operations Research* , vol. 33, p. 1875–93, 2006.

[4] M. Wright, "Timetabling county cricket fixtures using a form of tabu search," *Journal of the Operational Research,* vol. 45, no. 7, pp. 758-770, 1994.

[5] J. Dinitz and D. Stinson, "On assigning referees to tournament schedules," *Bulletin of the Institute of Combinatorics and its Applications,* vol. 44, pp. 22-38, 2005.

[6] A. Duarte and C. Ribeiro, "Referee assignment in sports leagues: approximate and exact multiobjective approaches," in *19th international conference on multiple criteria decision making,* Aukland, 2008.

[7] G. Kendall, S. Knust, C. Riberio and S. Uruttia, "Scheduling in sports: An annotated bibliography," *Computers and Operations Research,* vol. 37, pp. 1-19, 2010.

[8] C. Ribeiro and S. Urrutia, "Heuristics for the Mirrored Traveling Tournament Problem," *European Journal of Operational Research*, vol. 179, pp. 775-787, 2007.

[9] V. A. Weert and J. Schreuder, "Construction of basic match schedules for sports competitions by using graph theory," in *The 2nd international conference on the practice and theory of automated timetabling, Lecture notes in computer science*, vol. 1408, E. Burke and M. Carter, Eds., Berlin, Springer, 1998, pp. 201-210.

[10] M. Trick, "Challenge Traveling Tournament Instances," [Online]. Available: http://mat.gsia.cmu.edu/TOURN/. [Accessed 10 May 2013].

[11] C. Riberio and S. Urrutia, "Maximizing breaks and bounding solutions to the mirrored traveling tournament problem," *Discrete Applied Mathematics,* vol. 154, pp. 1932-1948, 2006.

[12] K. Easton, G. Nemhauser and M. Trick, "Solving the Travelling Tournament Problem: A Combined Integer Programming and Constraint Programming Approach," in *Practice and Theory of Automated Timetabling IV, Lecture Noes in Computer Science*, vol. 2740, E. Burke and P. Causmaecker, Eds., Springer, 2003, pp. 100-109.

[13] T. Benoist, F. Laburthe and B. Rottembourg, "Lagrange relaxation and constraint programming collaborative schemes for traveling tournament problems," in *International Workshop on Integration of AI and OR Techniques*, Ashford, Kent, 2001.

[14] M. Hongwei, "Biogeography based optimization for Traveling Salesman Problem," in *International conference of natural computation*, Yantai, Shandong, 2010.

[15] y. Song, M. Liu and Z. Wang, "Biogeography-Based Optimization for the Traveling Salesman Problems," in *International Joint Conference on on Computational Science and Optimization (CSO)*, Huangshan, Anhui, China, 2010.

[16] R. Melo, S. Urrutia and C. Ribeiro, "The traveling tournament problem with predefined venues," *Journal of Scheduling*, vol. 12, no. 6, pp. 607-622, 2009.

[17] A. Anagnostopoulos, L. Michel, P. V. Hentenryck and Y. Vergados, "A simulated annealing approach to the traveling tournament problem," *Journal of Scheduling*, vol. 9, no. 2, pp. 177-193, 2006.

[18] F. Costa, S. Urrutia and C. Riberio, "An ILS heuristic for the traveling tournament problem with fixed venues," in *Proceedings of the 7th International conference on the practice and theory of automatic timetabling*, Montreal, 2008.

[19] S. Urrutia, C. Ribeiro and R. Melo, "A new lower bounds to the traveling tournament problem," in *IEEE symposium on computational intelligence in scheduling*, Honolulu, 2007.

[20] L. DiGaspero and A. Schaerf, "A composite-nighbourhood tabu search approach to the traveling tournament problem," *Journal of Heuristics*, vol. 13, pp. 189-207, 2007.

[21] A. Lim, B. Rodrigues and X. Zhang, "A simulated annealing and hill climbling algorithm for the traveling tournament problem," *European Journal of Operations Research,* vol. 174, pp. 1459-1478, 2006.

[22] P. V. Hentenryck and Y. Vergados, "Traveling tournament scheduling: A systematic evaluation of simulated annealing," in *Integration of AI and OR techniques in constraint programming for combinatorial optimization problems, Lecture Notes in computer science,* vol. 3990, Berlin, Springer, 2006, pp. 228-243.

[23] K. Cheung, "Solving mirrored traveling tournament problem benchmark instances with eight teams," *DiscreteOptimization,* vol. 5, pp. 138-143, 2008.

[24] F. Biajoli and L. Lorena, "Mirrored Traveling Tournament Problem: An Evolutionary Approach," in *Advances in Artificial Intelligence - IBERAMIA-SBIA 2006,* vol. 4140, Springer, 208-217, pp. 208-217.

[25] A. Tajbakhshl, K. Eshghi and A. Shamsi, "A hybrid PSO-SA algorithm for the Traveling Tournament Problem," in *International conference on Computers & Industrial Engineering*, Troyes, 2009.

[26] P.C. Chen , G. Kendall and G. Berghe, "An Ant Based Hyper-heuristic for the Travelling Tournament Problem," in *IEEE Symposium on Computational Intelligence in Scheduling*, Honolulu, HI, 2007.

[27] D. Simon, "Biogeography-Based Optimization," *IEEE Transactions on Evolutionary Computation,* vol. 12, no. 6, pp. 702 - 713, 2008.

[28] M. Haping, F. Minrui, D. Zhiguo and J. Jing, "Biogeography-based optimization with ensemble of migration models for global numerical

optimization," in *IEEE Congress on Evolutionary Computation (CEC)*, Brisbane, QLD, 2012.

[29] Y. Wang and C. Zhihua, "A novel hybrid biogeography-based optimization with differential mutation," in *International conference on Electronic and Mechanical Engineering and Information Technology (EMEIT)*, Harbin, Heilongjiang, China, 2011.

[30] D. Simon and D. Daweiu, "Oppositional biogeography-based optimization," in *IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, TX, 2009.

[31] D. Simon, M. Ergezer and D. Dawei, "Population distributions in biogeography-based optimization algorithms with elitism," in *IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, TX, 2009.

[32] S. Kirkpatrick, C. Gelatt and M. Vecchi, "Optimization by Simulated Annealing," *American Association for the Advancement of Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[33] G. Ridgway, "Matlab Website," Mathworks, [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/27321-unique-random-permutations. [Accessed 1 May 2013].

[34] J. Dinitz, E. Lamken and W. Wallis, "Scheduling a tournament," in *Handbook of Combinatorial Designs*, CRC Press, 1995, pp. 578-584.

[35] R. Poli, "Analysis of the Publications on the Applications of Particle Swarm Optimization," *Journal of Artificial Evolution and Applications,* pp. 1-10, 2008.

[36] D. Palupi Rini, S. M. Shamsuddin and S. S. Yuhaniz, "Particle Swarm Optimization: Technique, System and Challenges," *International Journal of Computer Applications,* vol. 14, no. 1, pp. 19-27, 2011.

[37] D. Simon, "Biogeography-Based Optimization," April 2014. [Online]. Available: http://embeddedlab.csuohio.edu/BBO/.