# SOLVING THE TRAVELING TOURNAMENT PROBLEM USING SIMULATED ANNEALING

BY

EDMOND ROBERT ZABLAN ORTAL

AN UNDERGRADUATE RESEARCH PAPER
SUBMITTED TO THE
DEPARTMENT OF MATHEMATICS
COLLEGE OF SCIENCE
THE UNIVERSITY OF THE PHILIPPINES
DILIMAN, QUEZON CITY

IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF SCIENCE IN MATHEMATICS

APRIL 2006

The University of the Philippines
College of Science
Department of Mathematics

This undergraduate research paper hereto attached, entitled SOLVING THE TRAVELING TOURNAMENT PROBLEM USING SIMULATED ANNEALING, prepared and submitted by **Edmond Robert Zablan Ortal**, in partial fulfillment of the requirements for the degree of **Bachelor of Science in Mathematics**, is hereby accepted.

Kimberly May M. Vallesteros
Adviser

Accepted and approved in partial fulfillment of the requirements for the degree of Bachelor of Science in Mathematics.

Jose Ernie C. Lope, Ph.D.
Chairman
Department of Mathematics

## Abstract

ORTAL, EDMOND ROBERT ZABLAN. SOLVING THE TRAVELING TOURNAMENT PROBLEM USING SIMULATED ANNEALING. (Under the direction of Kimberly May M. Vallesteros) This paper tackles the traveling tournament problem (TTP), which considers the important aspects in scheduling the US Major League Baseball (MLB) games. The problem is combinatorially challenging even for a small number of teams. Metaheuristics are often used to find "near-optimal" solution to various TTP instances. Simulated Annealing (SA), a metaheuristic method, is an iterative, stochastic algorithm that is analogous to the concept of physical annealing. We consider the case of a 6-team MLB-National League, referred to as NL6 instance of TTP, presented in http://mat.gsia.cmu.edu/TOURN. A modified version of the SA algorithm presented in the paper "A Simulated Annealing Approach to the Traveling Tournament Problem" by Anagnostopoulos, et. al., is implemented using Visual Basic. This program was ran several times with the parameter values changed in each run. Further modifications were then made to the program. Finally, after obtaining and analyzing the results, recommendations were given to further improve the process of obtaining best possible solutions even in larger TTP instances.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Sports leagues are big businesses around the world for they represent significant sources of revenue for many sectors in the society. State-of-the-art facilities are used in the staging of games and are being flocked by many fans and aficionados. Team owners get their share of the revenue from gate receipts. Prominent players get to endorse various products, in addition to signing multimillion-dollar contracts. Many advertisers buy significant amount of airtime when the games are broadcast either on TV, the radio or even the Internet ([KE], [CR2], [PVH]).

Arguably the most crucial activity in the operations of any sports league is the scheduling of games. After all, games are the heart and soul of these enterprises. In many leagues however there are lots of requirements, preferences and idiosyncrasies that need to be satisfied, thus affecting the scheduling process [MT4]. The Major League Baseball (one of the biggest and the oldest professional sports leagues in the United States), for instance, has hundreds of pages of requirements and requests that complicate the scheduling of games. In addition, the said league is composed of 30 teams (with 29 of them scattered throughout the United States and 1 team playing in Canada) divided into two leagues. Every year, each team plays 162 games, half of which is played in their respective home courts while the rest are played in an opponent's facility. Of course, the minimization of entailed costs in staging games, as well as the appeal of matchups to the audience and advertisers, is taken into consideration as the schedule is being created.

We can see that the Major League Baseball, like many prominent sports leagues across the globe, has a challenging scheduling process which it undergoes every year. In an attempt to simplify in particular the MLB scheduling, a problem class has been proposed in [KE] and considers the key issues that may characterize a typical sports league in general. The traveling tournament problem (TTP), as it is called, considers the conflict between distance traveled and the home/away assignment in each game [AA].

The TTP, as described in [LD], is a combinatorial problem that exhibits issues concerning both feasibility (with respect to structuring a tournament) and optimization (regarding the traveling distance). Although the conditions in a typical league scheduling have been simplified with the proposition of the TTP, creating an optimal schedule is still difficult for nontrivial instances. While solving the two issues separately is rather easy, a combination of these two makes the problem difficult [AA]. [GTL] classifies the problem as NP-hard. Aside from difficulty in solving it using exact methods, considering all the possible solution will overwhelm any computer with the space requirement and the inconceivable amount of time it consumes ([LD], [GTL]). As a result, researchers seek to develop methods which can be easily implemented in creating schedules that have "near optimal" cost and at the same time satisfy a set of constraints.

The rest of this paper is divided as follows. Chapter 2 presents the formulation of the TTP. The implementation of the simulated annealing (SA) metaheuristic to our problem is discussed in the following chapter. We consider a particular scheduling problem, which is to be presented later. Using a program that the author has created using Visual Basic® for Excel (with the author referring to the [ME] and [VB] for help), we perform the SA algorithm in an attempt to find "good enough" solution to the existing problem. The results of this experiment, obtained through several runs made to the VB program, are shown in Chapter 4. A list of recommendations, which were based on the earlier experimentation, is given in Chapter 5.

# Chapter 2

# Formulation of the Traveling Tournament Problem

In this chapter, we first define the sets and variables important in the formulation of the TTP. Afterwards, the constraints and the objective function are discussed. The particular problem considered for this study is presented after that. Finally, the complete mathematical formulation for this specific problem is presented.

## 2.1 Definition of Sets and Variables

Let **T** be the set of teams in the tournament. For simplicity in the formulation, we let **T** = {1 … t}, instead of using actual team names. (In the specific problem we shall see later, the teams were represented by numbers.) We assume that |**T**| = t > 2 and is even.

Let **R** be the set of rounds (or slots). We also let **R** = {1 … 2(t-1)}, instead of using actual time periods. We assume that the schedule of games is temporary constrained [GN] or compact [MT3], i.e. every team plays exactly one game per round. We now use the terms "game" and "round" interchangeably.

Now let a, b be in **T** and k in **R**. We define the following variables:

1. Let $D_{ab}$ be the distance between the home courts of teams a and b. (We shall somewhat incorrectly speak of the distance between a and b.)
    - We assume that any team plays its home games in exactly one venue and that no two teams share the same home court.

$$D_{ab} = \begin{cases} 0, if\ a = b \\ > 0, if\ a \neq b \end{cases}$$

(2.1)

- Also, we assume that the distance between any two teams is constant and independent of the directions traveled.

$$D_{ab} = D_{ba}$$

(2.2)

We can verify that a matrix whose entry in the ith row and jth column is $D_{ij}$ is nonnegative, real symmetric. We shall see an example of such distance matrix later in this chapter.

2. Let $O_{ak}$ be the opponent of team a on round k.
   - It is understood that no team plays against itself.

$$O_{ak} \neq a$$

(2.3)

   - Also, the opponent of any team's opponent is the team itself.

$$O_{O_{ak},a} = a$$

(2.4)

3. Let $V_{ak}$ be the venue of team a's game on round k.
   Before we proceed, we make the following definition, which will be useful in the formulation of the objective function.

$$V_{a0} = V_{a,2t-1} = a$$

(2.5)

k = 0 can be considered as the round prior to the first round, the start of the tournament. Meanwhile, $k = 2t - 1 = 2(t - 1) + 1$ is the round after the final round of the tournament. The above definition simply tells us that any team travels from its home court in going to the venue of its first game and returns there after it has played its last game. The home court, in addition to it being the venue of home games, can be considered as a "depot" for teams since this is where they come from prior to the start of the tournament and return there after the tournament is over.

We now make some assumptions on the variable $V_{ak}$.

- $V_{ak}$ is a binary variable. After all, for any round, a team has only two choices as to where it plays.

$$V_{ak} = \begin{cases} a, if\ a\ plays\ at\ its\ home\ court\ for\ its\ k^{th}\ game \\ O_{ak}, if\ a\ plays\ at\ its\ opponent's\ home\ court \\ \quad for\ its\ k^{th}\ game \end{cases}$$

(2.6)

- It follows from above that:

$$V_{ak} = V_{O_{ak},k}$$

(2.7)

## 2.2 Constraints

For the sake of continuity in our discussion, we deviate from the usual method of presenting the objective function before the constraints that are to be satisfied. There are some concepts in the objective function that need to be explained in the constraints part so we first give the constraints of the TTP.

We first define the following decision variables:

1. Let $G_{abk}$ be the decision variable that answers the question "Does a plays b on round k?"

$$G_{abk} = \begin{cases} 1, if\ O_{ak} = b \\ 0, if\ O_{ak} \neq b \end{cases}$$

(2.8)

2. Meanwhile, let $H_{abk}$ answer the question "Does a plays b at its home court on round k?"

$$H_{abk} = \begin{cases} 1, if\ G_{abk} = 1\ and\ V_{ak} = a \\ 0, otherwise \end{cases}$$

(2.9)

The TTP has two classes of constraints: the hard constraints and the soft constraints, which are defined as follows:

1. Hard constraints

   These are the conditions that must be satisfied by any schedule (configuration or solution). We can think of the hard constraints as the set of requirements that are strictly enforced during the construction of a schedule [ASc].

   a. Double-round robin constraint

      Any two teams play each other exactly twice, with the venue of their matches alternated between their respective home courts.

      i.) a and b play each other exactly twice.

$$\sum_{k=1}^{2(t-1)} G_{abk} = 2$$

(2.10)

      ii.) a plays b exactly once in its home court.

$$\sum_{k=1}^{2(t-1)} H_{abk} = 1$$

(2.11)

   b. Round constraints

      These are constraints that must be satisfied in any round k.

      i.) a plays b iff b plays a.

$$G_{abk} = G_{bak}$$

(2.12)

      ii.) A team has exactly one opponent.

$$\sum_{b\ in\ T\{a\}} G_{abk} = 1$$

(2.13)

iii.) In any game, one team plays the other team in its home court.

$$H_{a,O_{ak},k} + H_{O_{ak},a,k} = 1$$

(2.14)

In [AA], a) is the only hard constraint mentioned in the discussion. The constraints in b) were included in this formulation because after all these conditions have to be satisfied by any solution.

2. Soft constraints

These conditions may be violated by any schedule. We can think of the soft constraints as the preferences or wishes of the audience, officials, advertisers, etc. regarding certain aspects concerning the schedule. As will be explained in the next chapter, a violation of any of these constraints subjects the schedule to a corresponding positive penalty [ASc]. In addition, such schedule is said to be infeasible [AA].

Due to the absence of a predictive model for attendance, which is a crucial element in any sports league, the soft constraints, when violated, can be considered as features that can be detrimental to game attendance [WCJ].

a. Atmost constraint

Let 1 ≤ s ≤ t-1. A team can have a home stand (string of consecutive home games) or road trip (consecutive road games) of at most length s.

For k = 1 to (2(t - 1) – s):

$$1 \le \sum_{r=k}^{k+s} H_{a,O_{ar},r} \le s$$

(2.15)

The variable s is referred to in [KE] as the integer parameter. This serves as a measure of tradeoff between distance traveled and the length of the home stands and road trips. This also measures the tradeoff between the welfare of the players and the interest of the audience.

A small value of s, for instance, results to an increased travel distance. The TTP, in effect, resembles the vehicle routing problem because teams have to return home often and immediately after a short road trip. The build-up of audience interest back home tends to be slow. Also it causes inconvenience in the part of the players who have to travel often.

On the other hand, a rather large value of s tends to deplete audience interest (for teams having a long road trip), as well as finances (for long home stands). This time, the problem resembles the traveling salesman problem ([WCJ], [KE]).

We now define the variable AV. AV refers to the number of occurrences (i.e. the number of times for any team and any round) that a home stand or road trip exceeds s consecutive games. The round when the violation started is considered in the count. Note that a home stand (or road trip) with length s' counts as (s'–s) violation(s).

$$AV = \left|\left\{(a,k)\colon \sum_{r=k}^{k+s} H_{a,O_{ar},r} = s+1\right\}\right| + \left|\left\{(a,k)\colon \sum_{r=k}^{k+s} H_{a,O_{ar},r} = 0\right\}\right|$$

(2.16)

b. Norepeat constraint

No two teams play each other for consecutive rounds.

For k = 1 to (2t - 3):

$$G_{a,O_{ak},k+1} = 0$$

(2.17)

The variable NV refers to the number of occurrences wherein a team plays another team for consecutive rounds.

$$NV = \left|\{(a,k): G_{a,O_{ak},k+1} = 1\}\right| \tag{2.18}$$

Note that NV is always even.

## 2.3 Objective Function

Our objective is to minimize the total traveling distance of all the teams for the entire tournament. However, a minimum distance may, for instance, be due to teams having a long home stand. The existence of many violations makes a schedule unattractive. To discourage the selection of infeasible schedules as optimal schedules, a penalty function P(v) is imposed. The objective function then is defined as follows:

$$\min TC = (TD^2 + [w \cdot P(v)]^2)^{\frac{1}{2}} \tag{2.19}$$

where $TD =$ total distance traveled by all teams for the entire tournament

$$= \sum_{a=1}^{t} \sum_{k=0}^{2(t-1)} D_{V_{ak},V_{a,k+1}} \tag{2.20}$$

$w = weight\ of\ penalty\ (which\ has\ the\ same\ unit\ as\ distance)$

$$v = total\ number\ of\ violations = AV + NV \tag{2.21}$$

and $P(v) = cost\ of\ penalty\ for\ v\ violations$

The objective function is defined as above so that while the penalty contributes to the total cost, it does not have a dominating effect over the total distance.

P(v) is a sublinear function such that P(1) = 1. Such function is chosen with the intuition that the first violation costs more than the subsequent ones. After all, the first violation made a schedule cross the line from feasibility to infeasibility. It was experimentally decided that $P(v) = 1 + \frac{v^{1/2} \ln v}{2}$ so that the penalty function value does not grow too slowly to avoid schedules with a severe degree of infeasibility [AA]. Figure 2.1 shows us the graph of P(v).

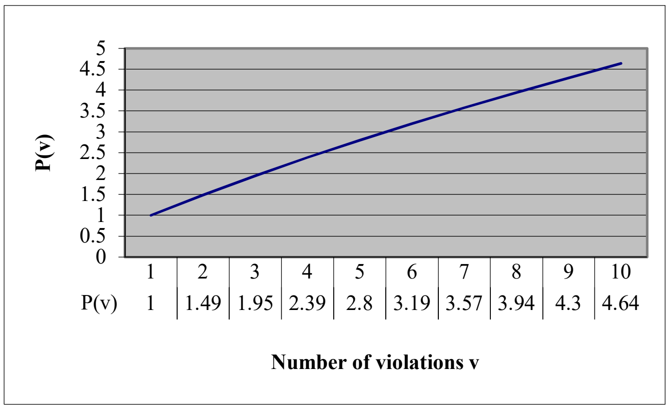

**Figure 2.1:** Graph of the sublinear function P(v)

## 2.4 Discussion Regarding the Specific Problem Considered

For this research, we consider the NL6 instance of the TTP, a problem posted in [MT2]. We are given 6 teams from the National League of the US MLB. We are also given the distance matrix between any of these 6 teams.

| | | |
|---|---|---|
| **1** | Atlanta Braves | ATL |
| **2** | New York Mets | NYM |
| **3** | Philadelphia Phillies | PHI |
| **4** | Montreal Expos | MON |
| **5** | Florida Marlins | FLA |
| **6** | Pittsburgh Pirates | PIT |

**Table 2.1:** NL6 teams

Table 2.1 shows the 6 teams considered in this problem. Each team is represented by a number. The number 1 corresponds to the Atlanta Braves, for instance. Such representation helps facilitate our formulation for this specific problem.

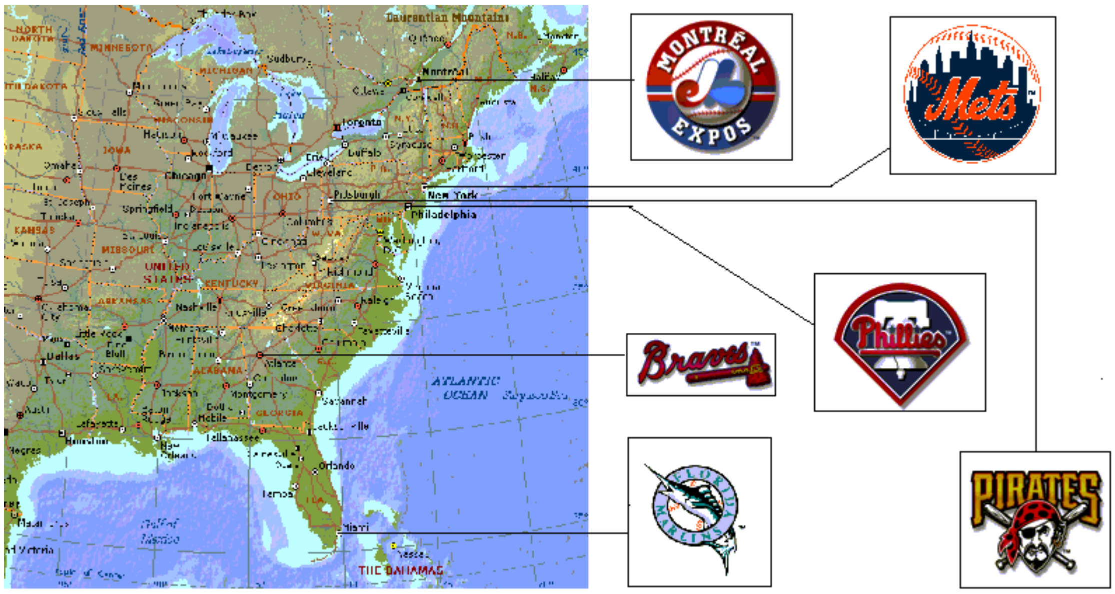

**Figure 2.2:** A portion of the North American map and the NL6 team logos (Map courtesy of http://encarta.msn.com/map_701515164/North_America.html, team logos courtesy of http://en.wikipedia.org/wiki/)

| | **1** | **2** | **3** | **4** | **5** | **6** |
|---|---|---|---|---|---|---|
| **1** | 0 | 745 | 665 | 929 | 605 | 521 |
| **2** | 745 | 0 | 80 | 337 | 1090 | 315 |
| **3** | 665 | 80 | 0 | 380 | 1020 | 257 |
| **4** | 929 | 337 | 380 | 0 | 1380 | 408 |
| **5** | 605 | 1090 | 1020 | 1380 | 0 | 1010 |
| **6** | 521 | 315 | 257 | 408 | 1010 | 0 |

**Table 2.2:** The distance matrix for NL6 instance

Figure 2.2 shows the locations of the NL6 teams, which are represented by their team logos, in the North American map. Observe that these teams, with the exception of the Montreal Expos (which by the way is now known as the Washington Nationals), the location of these teams are mostly concentrated in the US East coast. Meanwhile, Table 2.2 gives us the matrix representation of the actual air distances of the cities represented in the NL6 instance. As expected, this matrix is nonnegative, real symmetric [MT2].

The schedule for this problem is represented by a 6 X 10 matrix, as shown in Table 2.3. The entry in the i$^{th}$ row and the j$^{th}$ column represents the opponent of team i on round k, as well as the venue where the said team plays at that round. If the entry in that particular location is preceded by "@," then the said team plays at its opponent's home court for that round. Otherwise, the team plays at its home court. We can verify that the schedule in Table 2.3 satisfies the hard constraints discussed earlier.

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 6 | 3 | 5 | @2 | @5 | @3 | 2 | @4 | @6 | 4 |
| **2** | @5 | @6 | @3 | 1 | 6 | @4 | @1 | 5 | 4 | 3 |
| **3** | @4 | @1 | 2 | @6 | 4 | 1 | @5 | 6 | 5 | @2 |
| **4** | 3 | 5 | 6 | @5 | @3 | 2 | @6 | 1 | @2 | @1 |
| **5** | 2 | @4 | @1 | 4 | 1 | @6 | 3 | @2 | @3 | 6 |
| **6** | @1 | 2 | @4 | 3 | @2 | 5 | 4 | @3 | 1 | @5 |

**Table 2.3:** Matrix representation of a schedule

In addition, s=3 for this problem. This means that it is preferred for a schedule not to have more than 3 consecutive home (or road) games. We can see that Table 2.3 satisfies both the atmost and norepeat constraints. Hence, the said schedule is feasible.

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | ~~6~~ | ~~3~~ | ~~2~~ | ~~5~~ | ~~@5~~ | ~~@3~~ | ~~@2~~ | ~~@4~~ | ~~@6~~ | 4 |
| **2** | ~~@5~~ | ~~@6~~ | ~~@1~~ | ~~@3~~ | 6 | @4 | ~~1~~ | ~~5~~ | 4 | ~~3~~ |
| **3** | @4 | @1 | @6 | 2 | 4 | 1 | @5 | 6 | 5 | @2 |
| **4** | 3 | **5** | **@5** | 6 | @3 | 2 | @6 | 1 | @2 | @1 |
| **5** | 2 | **@4** | **4** | @1 | 1 | @6 | 3 | @2 | @3 | 6 |
| **6** | @1 | 2 | 3 | @4 | @2 | 5 | 4 | @3 | 1 | @5 |

**Table 2.4:** An infeasible schedule

Table 2.4 is an infeasible schedule. Although it does satisfy the hard constraints, we can see that it has violated both the atmost and norepeat constraints. The entries in boldface are those matchups that violated the norepeat constraint. Meanwhile, those entries that bear the strikethrough effect are those that violated the atmost constraint. The 5 consecutive road games that team 1 have count as 2 violations. All in all, Table 2.4 contains 7 violations.

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **Home** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 6 | 3 | 5 | @2 | @5 | @3 | 2 | @4 | @6 | 4 | **1** | |
| **2** | @5 | @6 | @3 | 1 | 6 | @4 | @1 | 5 | 4 | 3 | **2** | |
| **3** | @4 | @1 | 2 | @6 | 4 | 1 | @5 | 6 | 5 | @2 | **3** | |
| **4** | 3 | 5 | 6 | @5 | @3 | 2 | @6 | 1 | @2 | @1 | **4** | |
| **5** | 2 | @4 | @1 | 4 | 1 | @6 | 3 | @2 | @3 | 6 | **5** | |
| **6** | @1 | 2 | @4 | 3 | @2 | 5 | 4 | @3 | 1 | @5 | **6** | |
| | | | | | | | | | | | | **TD** |
| **1** | 0 | 0 | 0 | 745 | 1090 | 1020 | 665 | 929 | 408 | 521 | 0 | 5378 |
| **2** | 1090 | 1010 | 257 | 80 | 0 | 337 | 929 | 745 | 0 | 0 | 0 | 4448 |
| **3** | 380 | 929 | 665 | 257 | 257 | 0 | 0 | 0 | 1020 | 1090 | 80 | 4678 |
| **4** | 0 | 0 | 0 | 1380 | 1020 | 380 | 408 | 408 | 337 | 745 | 929 | 5607 |
| **5** | 0 | 1380 | 929 | 605 | 0 | 1010 | 257 | 80 | 1090 | 0 | 0 | 5351 |
| **6** | 521 | 521 | 408 | 408 | 315 | 315 | 0 | 257 | 257 | 1010 | 1010 | 5022 |
| | | | | | | | | | | | | 30484 |

**Table 2.5:** Computing the total distance traveled

In Table 2.5, we can see how the total distance is computed. For instance, team 1's game in round 1 is held at its own home court. Hence, the distance traveled in that round is 0 since it didn't travel at all in going to the venue of that game coming from the venue of the previous game (held in round 0, which is not a game at all). It is in this aspect that our definition in Section 2.1 regarding home courts serving as "depot" for teams becomes useful.

## 2.5 The Complete Mathematical Formulation of the Specific Problem Considered

$\min TC = (TD^2 + [w \cdot P(v)]^2)^{\frac{1}{2}}$, subject to:

For a, b in **T** and k in **R**:

<u>Decision Variables</u>

$$G_{abk} = \begin{cases} 1, if\ O_{ak} = b \\ 0, if\ O_{ak} \neq b \end{cases}$$

$$H_{abk} = \begin{cases} 1, if\ G_{abk} = 1\ and\ V_{ak} = a \\ 0, otherwise \end{cases}$$

<u>Hard Constraints</u>

a. Double-round robin constraints

$$\sum_{k=1}^{2(t-1)} G_{abk} = 2$$

$$\sum_{k=1}^{2(t-1)} H_{abk} = 1$$

b. Round constraints

$$G_{abk} = G_{bak}$$

$$\sum_{b\ in\ T\{a\}} G_{abk} = 1$$

$$H_{a,O_{ak},k} + H_{O_{ak},a,k} = 1$$

<u>Soft Constraints</u>

a. Atmost constraint

$$1 \le \sum_{r=k}^{k+s} H_{a,O_{ar},r} \le 3, for\ k = 1\ to\ 7$$

b. Norepeat constraint

$$G_{a,O_{ak},k+1} = 0, for\ k = 1\ to\ 9$$

# Chapter 3

# Implementation of the Simulated Annealing on the Traveling Tournament Problem

In this chapter, we first discuss a brief background on the previous works done involving scheduling of games. Afterwards, the bulk of this chapter will focus on the discussion of the Simulated Annealing metaheuristic. First we give a brief background on metaheuristics. Afterwards, we discuss the background on SA. Then we give an outline on how SA is implemented in finding solutions for TTP. Finally, we discuss the design of the program done in Visual Basic® for Excel which was used to implement the SA algorithm.

## 3.1 Previous Works in Sports Scheduling

As mentioned in Chapter 1, the TTP is a NP-hard problem. Hence finding a method for generating good solutions to this problem has become the pursuit of some researchers over the past 30 years. Examples of methods used in finding solutions for TTP (and sports scheduling in general) were tabu search (see [LD], [LM] and [JH]), genetic algorithm (see [JL], whose study centered on the National Basketball League in Australia), ant algorithm (see [HC], whose study on generate-and-test heuristic was inspired by this, and [MA], whose study on agent-based metaheuristic was also inspired by this algorithm), constructive heuristic (see [CR1], whose work focused on Brazilian soccer leagues. Cesar Ribeiro and Sebastián Urrutia also developed a computer system based on the GRASP metaheuristic [CR2].), network structures (see [ASu]), and simulated annealing (see [PVH], [FB], whose study focused on creating schedules for the Brazilian Soccer Championship and [AA]). In addition, some early works in scheduling were [WCJ], which presented a computer-assisted heuristic, and [RC], whose approach is a modification of the Balas' implicit enumeration. More recent works in this field involved the use of integer programming (see [GN], whose paper focused on creating schedule of men's basketball games for Atlantic Coast Conference of NCAA-Division I, [Trick, SSIP], [MT1]), constrained programming (see [GTL], [MH] – whose work revisited the study done in [GN], [TB] – which is about CP collaborating with Lagrangian relaxation, and [Trick, IPCP] – whose approach to scheduling is the combined efforts of IP and CP).

In this paper, we focus on simulated annealing as an approach to finding solutions to the TTP. The algorithm, which will be outlined later, was implemented in a program that was created using Visual Basic® for Excel.

## 3.2 Brief Background on Metaheuristics

As mentioned in Chapter 1, finding good solutions for TTP is difficult for non-trivial cases. Hence metaheuristics are used in finding solutions for such problem. Metaheuristics are relatively recent methods that originated from heuristics.

A heuristic search, according to [EB], is a trial and error treatment in finding a "good-enough" solution to a problem wherein there is no algorithm which directly applies in solving it. Heuristics were developed with the realization that it is impractical to examine every possible solution of a non-trivial problem. It was said earlier that such practice would take an incomprehensible amount of time.

However these heuristic methods are problem-specific. A new method has to be developed to carefully and exactly fit to the problem being considered [FH]. Examples of heuristic methods are Clarke and Wright Savings algorithm, Sweep algorithm and Tour Partitioning heuristic [KV].

All these changed in the recent years with the development of powerful and versatile metaheuristics. These solution methods are flexible enough to be adapted into various problems. They also generate higher quality solutions compared to heuristic methods. They create a link between local improvement methods (resemble that of hill-climbing procedure) and higher-level strategies so that a process, which is able to escape local optima and explore a wide range of the solution space, is created ([FH], [KV]). Some of the more common metaheuristics are tabu search, genetic algorithm and simulated annealing.

## 3.3 Brief Background on Simulated Annealing

Simulated annealing is an iterative algorithm that is analogous to the concept of physical annealing process. A glass or metal is initially melted at a high temperature and is slowly cooled until a point is reached when the properties of the substance is desirable and is stable. At any temperature value T during the process, the energy level E of the atoms is fluctuating but tending to decrease. Changes in the fluctuation of the energy level occur randomly and it is only occasionally that configurations that increase the energy level are accepted with probability $P(\Delta E) = e^{[-\Delta E/(k_B \cdot T)]}$, where $k_B$ = Boltzmann's constant ([FH], [KV]).

E is analogous to the cost of the solution. The transition from one energy level to another corresponds to the iteration. A sequence of these transitions for a given temperature value can be thought of as the phase, while the number of transitions before the temperature is lowered to its next value is referred to as the phase length. Meanwhile, the configurations obtained refer to the solutions obtained in the SA algorithm. As the phase length increases to infinity, the probability distribution of the states must approach the stationary distribution. This is a guarantee that SA will converge and thus the optimal solution can be found if we just keep on searching for the solution. However, in a finite phase length, the best solution obtained is referred to as quasi-equilibrium ([KV], [PVH]).

In any iteration, the algorithm seeks to improve the solution it has found thus far. However, solutions that worsen the optimal solution found are sometimes accepted, with the same probability mentioned earlier. This just shows that SA is also a probabilistic algorithm, which accommodates some solutions that worsen the objection function value so that the solution space being explored is expanded.

Perhaps the most crucial element in SA algorithm is the temperature. It measures the tendency of accepting a worse solution at any time during the run of the algorithm. Hence a temperature-cooling scheme has to be carefully planned. This will be discussed in the next section.

SA has been used to solve many optimization problems, some of which are the traveling salesman problem (see [FH] for an example), timetabling (see [JP]), graph problems, facility layout problem, vehicle routing problem (see [KV]), and even nonlinear programming (see [FH]).

## 3.4 The Simulated Annealing Algorithm

We now discuss how SA is implemented on TTP using the outline presented in [FH]. There are 4 general stages in the SA implementation.

1. Initialization

   We select any double-round robin schedule as the initial trial solution. Its corresponding cost is assigned as the initial trial cost. Initially, the initial trial solution and the initial trial cost will serve as our accepted solution so far and accepted cost so far.

   We also set the value of the SA parameters. The parameters are as follows:

   - $T_0$ = initial temperature
   - r = reduction factor, where $0 \leq r \leq 1$

     The reduction factor is a fixed value that is multiplied to the current temperature value to lower it. This is also known as the cooling factor.
   - w = weight of penalty (see discussion in Section 2.3)
   - temperature-lowering parameters
     - phase length (i.e. the number of iterations per temperature value)

- number of acceptances
- number of deteriorates (referred to as counter in [AA])

- algorithm-terminating parameters

- total number of rejections (for the entire SA run)
- total number of iterations (also for the entire run)
- desired final temperature value
- desired CPU time for SA run

The significance of these parameters is discussed as we move along in this outline.

2. Iteration

We randomly select a move that will generate the neighbor of our current trial solution. The neighborhood moves employed for this problem are as follows [AA]:

a. SwapHomes

This move performs an exchange of home/away assignments between two randomly selected teams in the rounds when these teams meet.

Suppose teams 2 and 5 are the randomly selected teams for this move, as seen in Table 3.1. In the table, the entries highlighted with a lighter shade are those occurrences wherein these two teams play each other. We see that these two teams meet it both rounds 3 and 5, with team 2 playing at team 5's home court in round 3 and team 5 playing at team 2's home court in round 5. The home/away assignments are swapped, as seen in Table 3.2.

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 5 | 2 | 6 | @3 | @4 | @6 | 3 | 4 | @2 | @5 |
| **2** | @6 | @1 | @5 | 4 | 5 | @3 | @4 | 6 | 1 | 3 |
| **3** | @4 | 5 | 4 | 1 | @6 | 2 | @1 | @5 | 6 | @2 |
| **4** | 3 | 6 | @3 | @2 | 1 | 5 | 2 | @1 | @5 | @6 |
| **5** | @1 | @3 | 2 | 6 | @2 | @4 | @6 | 3 | 4 | 1 |
| **6** | 2 | @4 | @1 | @5 | 3 | 1 | 5 | @2 | @3 | 4 |

**Table 3.1:** Before SwapHomes

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 5 | 2 | 6 | @3 | @4 | @6 | 3 | 4 | @2 | @5 |
| **2** | @6 | @1 | 5 | 4 | @5 | @3 | @4 | 6 | 1 | 3 |
| **3** | @4 | 5 | 4 | 1 | @6 | 2 | @1 | @5 | 6 | @2 |
| **4** | 3 | 6 | @3 | @2 | 1 | 5 | 2 | @1 | @5 | @6 |
| **5** | @1 | @3 | @2 | 6 | 2 | @4 | @6 | 3 | 4 | 1 |
| **6** | 2 | @4 | @1 | @5 | 3 | 1 | 5 | @2 | @3 | 4 |

**Table 3.2:** After SwapHomes

b. SwapRounds

This move performs an exchange of opponents between two randomly selected rounds. Suppose the two randomly selected rounds are 3 and 7, as seen from Table 3.3. Then the games in round 3 are transferred to round 7, and vice-versa, as seen in Table 3.4.

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 5 | 2 | 6 | @3 | @4 | @6 | 3 | 4 | @2 | @5 |
| **2** | @6 | @1 | @5 | 4 | 5 | @3 | @4 | 6 | 1 | 3 |
| **3** | @4 | 5 | 4 | 1 | @6 | 2 | @1 | @5 | 6 | @2 |
| **4** | 3 | 6 | @3 | @2 | 1 | 5 | 2 | @1 | @5 | @6 |
| **5** | @1 | @3 | 2 | 6 | @2 | @4 | @6 | 3 | 4 | 1 |
| **6** | 2 | @4 | @1 | @5 | 3 | 1 | 5 | @2 | @3 | 4 |

**Table 3.3:** Before SwapRounds

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 5 | 2 | 3 | @3 | @4 | @6 | 6 | 4 | @2 | @5 |
| **2** | @6 | @1 | @4 | 4 | @5 | @3 | @5 | 6 | 1 | 3 |
| **3** | @4 | 5 | @1 | 1 | @6 | 2 | 4 | @5 | 6 | @2 |
| **4** | 3 | 6 | 2 | @2 | 1 | 5 | @3 | @1 | @5 | @6 |
| **5** | @1 | @3 | @6 | 6 | 2 | @4 | 2 | 3 | 4 | 1 |
| **6** | 2 | @4 | 5 | @5 | 3 | 1 | @1 | @2 | @3 | 4 |

**Table 3.4:** After SwapRounds

c. SwapSked

This move performs an exchange of opponents between two randomly selected teams in every round except when these teams meet. Suppose Teams 1 and 6 are the two randomly selected teams, as seen in Table 3.5. Then, no swapping of entries is performed in rounds 3 and 6 since these are the rounds when they meet. Note that, in each of the other rounds, 4 entries are swapped. Those 4 affected teams are the two selected teams and their respective opponents in those rounds. We see this in Table 3.6, where the other affected entries bear the strikethrough effect.

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 5 | 2 | 6 | @3 | @4 | @6 | 3 | 4 | @2 | @5 |
| **2** | @6 | @1 | @5 | 4 | 5 | @3 | @4 | 6 | 1 | 3 |
| **3** | @4 | 5 | 4 | 1 | @6 | 2 | @1 | @5 | 6 | @2 |
| **4** | 3 | 6 | @3 | @2 | 1 | 5 | 2 | @1 | @5 | @6 |
| **5** | @1 | @3 | 2 | 6 | @2 | @4 | @6 | 3 | 4 | 1 |
| **6** | 2 | @4 | @1 | @5 | 3 | 1 | 5 | @2 | @3 | 4 |

**Table 3.5:** Before SwapSked

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 2 | @4 | 6 | @5 | 3 | @6 | 5 | @2 | @3 | 4 |
| **2** | ~~@1~~ | ~~@6~~ | @5 | 4 | 5 | @3 | @4 | ~~1~~ | ~~6~~ | 3 |
| **3** | @4 | 5 | 4 | ~~6~~ | ~~@1~~ | 2 | ~~@6~~ | @5 | ~~1~~ | @2 |
| **4** | 3 | ~~1~~ | @3 | @2 | ~~6~~ | 5 | 2 | ~~@6~~ | @5 | ~~@1~~ |
| **5** | ~~@6~~ | @3 | 2 | ~~1~~ | @2 | @4 | ~~@1~~ | 3 | 4 | ~~6~~ |
| **6** | 5 | 2 | @1 | @3 | @4 | 1 | 3 | 4 | @2 | @5 |

**Table 3.6:** After SwapSked

The 3 moves just discussed are not sufficient in exploring much of the schedules that can be possibly generated. The SwapHomes move can only consider two teams at a time and relying on this move will only slow down the search for best possible solutions. Meanwhile, there are certain nooks and crannies that the SwapRounds and SwapSked moves can't reach because they are "global" in nature. The SwapRounds move considers two rounds at a time and is inflexible because all the entries in those rounds are affected. The same scenario is true for SwapSked move. Hence, employing only these moves can cause the algorithm to be stuck in a local optimum, especially for large instances [AA]. This rationale justifies the need for more local moves, i.e.

those moves that can consider only a section of, say, a round. These local moves widen the space that can be explored. These are the two local moves presented in [AA]:

d. PartialSwapRounds

This move performs, on a randomly selected team, an exchange of games occurring between two randomly selected rounds. Suppose team 3, rounds 3 and 6 are the randomly selected items (see Table 3.7). Then the entries that are shaded lightly are the opponents of team 3 in the said rounds. However, merely swapping these entries will not give us a schedule that satisfies the hard constraints. Hence the schedule is adjusted deterministically.

In general, the PartialSwapRounds move generates the same results as SwapRounds. The key to generating the result seen in Table 3.8, i.e. all but two entries for each of the rounds involved are affected, and in effect enlarging the explored space is the selection of rounds. The randomly selected rounds must have a common game, i.e. the two rounds have common entries [AA].

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 5 | 2 | 6 | @3 | @4 | @6 | 3 | 4 | @2 | @5 |
| **2** | @6 | @1 | @5 | 4 | 5 | @3 | @4 | 6 | 1 | 3 |
| **3** | @4 | 5 | 4 | 1 | @6 | 2 | @1 | @5 | 6 | @2 |
| **4** | 3 | 6 | @3 | @2 | 1 | 5 | 2 | @1 | @5 | @6 |
| **5** | @1 | @3 | 2 | 6 | @2 | @4 | @6 | 3 | 4 | 1 |
| **6** | 2 | @4 | @1 | @5 | 3 | 1 | 5 | @2 | @3 | 4 |

**Table 3.7:** Before PartialSwapRounds

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 5 | 2 | 6 | @3 | @4 | @6 | 3 | 4 | @2 | @5 |
| **2** | @6 | @1 | ~~@3~~ | 4 | 5 | ~~@5~~ | @4 | 6 | 1 | 3 |
| **3** | @4 | 5 | 2 | 1 | @6 | 4 | @1 | @5 | 6 | @2 |
| **4** | 3 | 6 | ~~5~~ | @2 | 1 | ~~@3~~ | 2 | @1 | @5 | @6 |
| **5** | @1 | @3 | ~~@4~~ | 6 | @2 | ~~2~~ | @6 | 3 | 4 | 1 |
| **6** | 2 | @4 | @1 | @5 | 3 | 1 | 5 | @2 | @3 | 4 |

**Table 3.8:** After PartialSwapRounds

e. PartialSwapSked

This move performs, on a randomly selected round, an exchange of opponents between two randomly selected teams. In Table 3.9, teams 2 and 5 are selected, while round 10 is the selected round. The entries with lighter background in Table 3.9 are the respective opponents of teams 2 and 5 in round 10. We can see from Table 3.10 that 4 entries each from 4 rounds are affected.

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 5 | 2 | 6 | @3 | @4 | @6 | 3 | 4 | @2 | @5 |
| **2** | @6 | @1 | @5 | 4 | 5 | @3 | @4 | 6 | 1 | 3 |
| **3** | @4 | 5 | 4 | 1 | @6 | 2 | @1 | @5 | 6 | @2 |
| **4** | 3 | 6 | @3 | @2 | 1 | 5 | 2 | @1 | @5 | @6 |
| **5** | @1 | @3 | 2 | 6 | @2 | @4 | @6 | 3 | 4 | 1 |
| **6** | 2 | @4 | @1 | @5 | 3 | 1 | 5 | @2 | @3 | 4 |

**Table 3.9:** Before PartialSwapSked

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 5 | 2 | 6 | @3 | @4 | @6 | 3 | 4 | ~~@5~~ | ~~@2~~ |
| **2** | @6 | @1 | @5 | ~~6~~ | 5 | @3 | @4 | ~~3~~ | ~~4~~ | 1 |
| **3** | @4 | 5 | 4 | 1 | @6 | 2 | @1 | ~~@2~~ | 6 | ~~@5~~ |
| **4** | 3 | 6 | @3 | ~~@5~~ | 1 | 5 | 2 | @1 | ~~@2~~ | @6 |
| **5** | @1 | @3 | 2 | ~~4~~ | @2 | @4 | @6 | ~~6~~ | ~~1~~ | 3 |
| **6** | 2 | @4 | @1 | ~~@2~~ | 3 | 1 | 5 | ~~@5~~ | @3 | 4 |

**Table 3.10:** After PartialSwapSked

The cost of the schedule resulting from the selected move is computed. We then make use of the move selection rule, described as follows:

First, we let $Z_a$ = cost of the accepted schedule so far

$Z_c$ = cost of the current schedule

T = current temperature value

We then obtain the probability of acceptance, i.e. the probability that the current schedule will be accepted as the new best schedule so far.

$$Probability\ \{acceptance\} = \begin{cases} 1, if\ Z_c \leq\ Z_a \\ p =\ e^{\left[\frac{(Z_a - Z_c)}{T}\right]}, if\ Z_c >\ Z_a \end{cases}$$

(3.1)

The current schedule is always accepted when its corresponding cost is at most $Z_a$. Otherwise, it is accepted only when a randomly selected number between 0 and 1, exclusive, is less than or equal to p. Such acceptance is referred to as deterioration because the accepted schedule has a corresponding cost that is in fact worse than the previous $Z_a$. If the random number is greater than p, the current schedule is rejected and is discarded.

Based on (3.1), we can deduce that deteriorations abound when T is high. For a low T, rejections abound because as T approaches 0, the p also tends to 0. Only very few random numbers generated will be less than p.

This process is repeated in the next iteration and is applied to the newly accepted solution.

3. Check the temperature schedule

   The temperature schedule is the scheme used in gradually lowering the temperature value T. The temperature is lowered when any of the following occurs:

   - The desired phase length has been realized.
   - The desired number of acceptances has been attained.
   - The desired number of deteriorations has been achieved.

   In [KV], various temperature-lowering schemes are discussed. For our specific problem, we use the following scheme:

$$T_{new} = r \cdot T_{old}$$

(3.2)

   For a low value of r, fast cooling occurs. As a result, the algorithm tends to be stuck in a local optimum. A high value of r gives way to a rather slow cooling process. This leads to a long SA algorithm and a highly distorted iterative process because there is a tendency to have more deteriorates.

4. Stopping rule

   Finally the algorithm is terminated when any of the following occurs:

   - The desired total number of iterations has been performed.
   - The specified smallest value of temperature has been reached.
   - The total number of rejections has been achieved.
   - The CPU time has reached its specified limit.

   The schedule with the least cost is considered the optimal solution.

## 3.5 Design of the Program

For each run of the program created using Visual Basic® for Excel, a SA run report is generated as a Microsoft Excel worksheet. A sample report is shown in Table 3.11. Note that only the first 20 iterations of this particular run are shown. The iteration number is shown in the 1st column, while in the 2nd column the current temperature value is displayed. The 4th, 5th and 6th columns show the total distance, total penalty and total cost, respectively, of the schedule generated by the neighborhood move displayed in the 3rd column. Whether this schedule is accepted (or accepted as a deterioration) or rejected is known in the 7th column. If the schedule is a deterioration, the probability of acceptance, as determined by the formula in the previous section, is displayed. Finally, in the 8th column, the time it took to from the start of the algorithm to the current iteration is displayed.

| Iteration | Temperature | Move(s) Performed | Total Distance | Total Penalty | Total Cost | Remarks | Time |
|---|---|---|---|---|---|---|---|
| 0 | | | 31675.000 | 0.000 | 31675.000 | Initial sol'n | |
| 1 | 35000.000 | PartialSwapRounds | 31675.000 | 0.000 | 31675.000 | accepted | 0.938 |
| 2 | 35000.000 | PartialSwapSked | 31354.000 | 2980.258 | 31495.321 | accepted | 1.531 |
| 3 | 35000.000 | SwapRounds | 27263.000 | 7148.394 | 28184.583 | accepted | 1.844 |
| 4 | 35000.000 | SwapHomes | 28653.000 | 7148.394 | 29531.237 | 0.962 | 2.156 |
| 5 | 35000.000 | SwapRounds | 31099.000 | 7881.549 | 32082.185 | 0.930 | 2.453 |
| 6 | 35000.000 | PartialSwapRounds | 31740.000 | 9952.919 | 33263.917 | 0.967 | 2.734 |
| 7 | 35000.000 | SwapRounds | 27130.000 | 9952.919 | 28898.053 | accepted | 3.016 |
| 8 | 35000.000 | PartialSwapSked | 27012.000 | 9952.919 | 28787.302 | accepted | 3.453 |
| 9 | 35000.000 | SwapHomes | 27285.000 | 9281.413 | 28820.407 | 0.999 | 3.875 |
| 10 | 35000.000 | SwapSked | 31622.000 | 7148.394 | 32419.908 | 0.902 | 4.359 |
| 11 | 35000.000 | SwapRounds | 32849.000 | 5598.813 | 33322.718 | 0.975 | 4.781 |
| 12 | 35000.000 | SwapRounds | 28957.000 | 8591.674 | 30204.713 | accepted | 5.203 |
| 13 | 35000.000 | SwapSked | 28957.000 | 8591.674 | 30204.713 | accepted | 5.500 |
| 14 | 35000.000 | SwapHomes | 28665.000 | 7148.394 | 29542.880 | accepted | 5.813 |
| 15 | 35000.000 | SwapHomes | 28957.000 | 8591.674 | 30204.713 | 0.981 | 6.094 |
| 16 | 35000.000 | PartialSwapSked | 31728.000 | 9281.413 | 33057.686 | 0.922 | 6.469 |
| 17 | 35000.000 | SwapRounds | 32945.000 | 7881.549 | 33874.649 | rejected | 6.766 |
| 18 | 35000.000 | SwapRounds | 32473.000 | 10607.969 | 34161.744 | 0.969 | 7.203 |
| 19 | 35000.000 | SwapHomes | 33828.000 | 10607.969 | 35452.258 | 0.964 | 7.688 |
| 20 | 35000.000 | PartialSwapSked | 32547.000 | 7148.394 | 33322.766 | accepted | 8.203 |

**Table 3.11:** Sample SA run report

The algorithm is performed mostly in the “Intermediate” sheet, where the selected move is applied to the current schedule and afterwards the cost is computed. The information is then displayed in the SA run report. If the schedule is accepted, it is copied to the “Accepted” sheet. If the schedule in the “Intermediate” sheet is a deterioration, it is copied to the “Deteriorates” sheet. If rejected, the current schedule is discarded and the most recent accepted schedule is copied to the “Intermediate” sheet. At the end of the SA run, the obtained schedule with the minimum cost is displayed in the “Optimal” sheet while the best feasible schedule, if available, is displayed in the “Feasible” sheet.

The parameter values are crucial in determining the performance and the execution time of the SA run. In the program, the assigning of specific values to these parameters is done in the SA performance report (see Table 3.12).

| | |
|---|---|
| **Run #** | 1 |
| **Date** | 2/28/06 |
| **Time** | 1:59:29 PM |
| | |
| **PARAMETERS** | |
| **Number of teams** | 6 |
| **Maximum iterations** | 250 |
| **Initial temperature** | 700 |
| **Iterations per temperature** | 25 |
| **Reduction factor** | 0.9 |
| **Acceptances** | |
| **Rejections** | |
| **Deteriorations** | |
| **Minimum temperature** | |
| **Weight of penalty** | 2000 |

**Table 3.12:** SA performance report

The run #, date and time are automatically assigned every time the program is run. The values of the rest of the parameters are user-defined. The default number of teams is 6. The maximum number of iterations ranged from 100 to 2000, while the initial temperature value ranged from 100 to 20000. The maximum number of iterations per temperature value ranged from 20 to 30, while the reduction factor ranged from 0.7 to 0.999. The number of acceptances and deteriorations both ranged from 5 to 30, while the total number of rejections ranged from 100 to 2000. Preset final temperature value ranged from 1 to 100. Finally, the weight of penalty ranged from 1000 to 3000.

We can see from Table 3.12 that not all of the parameters were assigned with a value. It is because there are certain runs where the temperature-lowering condition is when the number of deteriorations has been reached and the algorithm-stopping condition is when the total number of rejections reached the pre-assigned value. In the study, not all the temperature-lowering and algorithm-stopping conditions are used simultaneously in a single run. Hence, there are certain parameters whose values are left unassigned for the particular run.

# Chapter 4

# Results and Discussion

In this chapter, the results of the SA runs done using the Visual Basic® program are presented. We first state the parameter values that were used in this problem. Afterwards some sample schedules that were generated are given and are compared to the best solution found so far to the NL6 instance. These results obtained are then analyzed. Afterwards, some modifications done to the program are presented. Some results generated are likewise presented.

## 4.1 SA Report and Sample Schedules Generated

In this section, we display the summary of some of the performed SA runs (Table 4.1). We focus the discussion on the elements under the "Results" heading. The numerical values in this part are iteratively generated during the execution of the program. The "Initial cost" is, of course, the cost of the initial solution, while the "Maximum cost" and "Minimum cost" are the maximum and minimum cost, respectively, obtained during the run. The "Average cost" and the "Standard deviation" are some statistical parameters that determine the dispersion of the costs obtained during the run. The "Number of iterations" refers to the last iteration performed during a run, while the "Number of acceptances," "Number of rejections," and the "Number of deteriorations" refer to the final count of acceptances, rejections and deteriorations, respectively.

| Run # | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **Date** | 2/28/06 | 2/28/06 | 2/28/06 | 3/3/06 | 3/3/06 |
| **Time** | 1:59:29 PM | 2:28:58 PM | 4:56:40 PM | 6:12:38 PM | 6:34:21 PM |
| | | | | | |
| **PARAMETERS** | | | | | |
| **Number of teams** | 6 | 6 | 6 | 6 | 6 |
| **Maximum iterations** | 250 | 250 | | | |
| **Initial temperature** | 700 | 800 | 500 | 35000 | 35000 |
| **Iterations per temperature** | 25 | 25 | 25 | 25 | 25 |
| **Reduction factor** | 0.9 | 0.95 | 0.9 | 0.9 | 0.95 |
| **Acceptances** | | | | | |
| **Rejections** | | | | | |
| **Deteriorations** | | | | | |
| **Minimum temperature** | | | 50 | 1 | 1 |
| **Weight of penalty** | 2000 | 2000 | 2000 | 2000 | 2000 |
| | | | | | |
| **RESULTS** | | | | | |
| **Initial cost** | 26181.644 | 29382.000 | 29330.000 | 31675.000 | 30594.000 |
| **Minimum cost** | 24958.412 | 25346.381 | 25974.994 | 22955.006 | 23684.827 |
| **Maximum cost** | 34371.005 | 35525.738 | 34248.732 | 37636.124 | 38705.788 |
| **Average cost** | 28515.643 | 29359.116 | 28565.434 | 28317.057 | 28339.572 |
| **Standard deviation** | 1861.514 | 1819.253 | 1698.012 | 3355.645 | 3206.030 |
| **Number of iterations** | 250 | 250 | 550 | 2500 | 5100 |
| **Number of acceptances** | 48 | 55 | 86 | 566 | 1151 |
| **Number of rejections** | 171 | 158 | 417 | 1515 | 3155 |
| **Number of deteriorations** | 31 | 37 | 47 | 419 | 794 |
| **Final temperature** | 271.194 | 504.200 | 54.709 | 1.033 | 1.052 |
| **Time (in seconds)** | 85.109 | 88.391 | 364.844 | 887.438 | 1827.063 |
| | | | | | |
| **Accepted feasible schedules** | 0 | 3 | 0 | 4 | 6 |
| | | | | | |
| **Cost of best feasible schedule** | #N/A! | 28404.000 | 29330.000 | 30594.000 | 29633.000 |
| **Time found (in seconds)** | #N/A! | 4.359 | 0 | 41.984 | 465.281 |
| **Iteration #** | #N/A! | 8 | 0 | 115 | 1261 |

**Table 4.1:** Summary of some SA runs

The "Final temperature" is the temperature value when the program is terminated while the "Time (in seconds)" refers to the length of time the program was executed. (Note that a Pentium IV processor was employed in the execution of most runs that were performed.) The "Accepted feasible schedules" is the number of penalty-free solutions that were accepted (either improves or worsens the currently accepted solution for that run) in the algorithm. If it is greater than zero, then the “Cost of best feasible schedule,” “Time found (in seconds),” and the “Iteration #” all have values.

Table 4.1 shows us some of the SA runs that were performed in the course of this study. Over a hundred runs were performed in the course of over a month but only 5 of these runs are presented and summarized in the table. We believe that these runs, as shown in the table, are representative of some possible scenarios that we shall explain shortly.

In Runs #1 and 2 (which were performed on 2/28/2006), the temperature-lowering parameter is the phase length (or the number of iterations per temperature value, while the algorithm-stopping parameter used is the total number of iterations in the entire run. We varied the initial temperature, reduction factor and the initial solution used (as seen in the initial cost part). Observe that the first two runs presented have the same value for almost all the parameter values under the “Results” heading except for the “Final temperature.” Observe also that in Run #1, there is no feasible schedule accepted. Thus the last 3 elements in Table 4.1 are not applicable. In Run #2, 3 feasible solutions were accepted. The best feasible cost is less than the initial cost for that run and is not equal to the “Minimum cost” obtained, and it was obtained 4 seconds into the execution of the program.

In these two runs, we see some possible scenarios involving the best solution. We see in Run #1 that there is no feasible schedule obtained, while in Run #2, we see that the best feasible schedule is obtained in the early part of the algorithm. Furthermore, in Run #2, we see that the schedule with the minimum cost is not necessarily the best schedule accepted in the algorithm.

Meanwhile, in runs #3, 4 and 5, the temperature is also lowered when the phase length desired has been reached. In these runs, however, the algorithm is terminated when the temperature value, when lowered, is smaller than the preset final temperature value. In Run #3, we used a rather low initial temperature and a rather high final temperature value. Runs #4 and 5 had different reduction factors. These 3 runs had different program-generated values for the parameters under the “Results” heading. Observe that these runs had a relatively long execution time. In Run # 3, no feasible schedule is accepted. Observe however that we used a feasible schedule as an initial solution. In Run #4, 4 feasible schedules were accepted. Observe that the cost of best feasible schedule is rather high and it was obtained in iteration #115. Meanwhile, 6 feasible schedules were obtained in Run #5. The best feasible schedule in that run was obtained more than 1000 iterations after the program started to run.

We see in Run #3 a case wherein the initial cost is that of a feasible schedule yet no feasible schedule was accepted in the algorithm. Run #4 portrays as scenario where the best feasible schedule has a cost that is a rather high value, while Run #5 is a scenario wherein the best feasible schedule is obtained after many iterations (and several temperature changes).

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **TOTAL COST** | 23916 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 5 | 2 | 6 | @3 | @4 | @6 | 3 | 4 | @2 | @5 | **Total distance** | 23916 |
| **2** | @6 | @1 | @5 | 4 | 5 | @3 | @4 | 6 | 1 | 3 | **Penalty** | 0 |
| **3** | @4 | 5 | 4 | 1 | @6 | 2 | @1 | @5 | 6 | @2 | | |
| **4** | 3 | @6 | @3 | @2 | 1 | 5 | 2 | @1 | @5 | 6 | **Violations** | 0 |
| **5** | @1 | @3 | 2 | 6 | @2 | @4 | @6 | 3 | 4 | 1 | **Repeats** | 0 |
| **6** | 2 | 4 | @1 | @5 | 3 | 1 | 5 | @2 | @3 | @4 | **Gaps** | 0 |

**Table 4.2:** Best feasible schedule so far for NL6 instance

We can see from Table 4.2 the best feasible schedule obtained so far for this particular instance. This was found by Kelly Easton on May 7, 1999 [MT2]. The best feasible schedules obtained in the SA runs performed in the study either were not obtained (thus non-existent) or were obtained but any of these scenarios occurred:

- The cost is either greater than that of the initial schedule and/or is not equal to the minimum cost obtained in the algorithm;
- It was obtained too early (as far as number of iterations and/or the temperature value are concerned) in the run;
- Its corresponding schedule is a neighbor to the schedule in Table 4.2 (i.e. that best feasible schedule is separated from the schedule in the said table by just one

neighborhood move). This was obtained last 2/22/2006. We see this schedule in Table 4.3.

- The said cost, in general, has a value that is not significantly different from the initial cost. This may mean that we got stuck in a local optimum.

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **TOTAL COST** | 23954 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 5 | 2 | 6 | @3 | @4 | @6 | 3 | 4 | @2 | @5 | **Total distance** | 23954 |
| **2** | @6 | @1 | @5 | 4 | 5 | @3 | @4 | 6 | 1 | 3 | **Penalty** | 0 |
| **3** | @4 | 5 | 4 | 1 | @6 | 2 | @1 | @5 | 6 | @2 | | |
| **4** | 3 | 6 | @3 | @2 | 1 | 5 | 2 | @1 | @5 | @6 | **Violations** | 0 |
| **5** | @1 | @3 | 2 | 6 | @2 | @4 | @6 | 3 | 4 | 1 | **Repeats** | 0 |
| **6** | 2 | @4 | @1 | @5 | 3 | 1 | 5 | @2 | @3 | 4 | **Gaps** | 0 |

**Table 4.3:** Best feasible schedule obtained in the study

## 4.2 Modifications to the Program

Due to the scenarios presented above, we decide to modify the SA program in the attempt to find a significantly better result. First, we restructured the objective of the problem. Instead of minimizing the cost, we minimize the penalty. In the event of a tie (which as we will see is a scenario), the schedule with the best cost is chosen. We did not explicitly state an objective function. We modified (3.1) so that the point of comparison will now be the penalty value and not the cost. This is advantageous in the sense that we can get the expected minimum value of this “objective function,” which is of course 0. Note that the penalty value P(v) (refer to Section 2.3), given a fixed weight of penalty, is a function of the number of violations. The number of violations is a discrete element since it can just be any of the nonnegative integers, of which the least value is zero. Also note that once a feasible solution is accepted, the cost of the schedule is taken into account as well. In effect, minimizing the penalty value is like a constraint satisfaction problem (CSP) [GTL] and multiobjective programming at the same time.

We see a performance summary for this reformulation in Table 4.4. For the runs, the temperature value is lowered when the assigned number of deteriorations has been reached.

The algorithm is terminated when the run reached a certain number of rejections. Observe that the number of acceptances greatly decreased and the number of feasible schedules accepted increased significantly. Since the objective now is minimization of the penalty value, once a feasible solution has been accepted it is quite hard to accept solutions that will be obtained later on, unless the schedule is feasible and has a cost better than that of the currently accepted feasible schedule. The number of deteriorations significantly increased as well, due to it being the temperature-lowering parameter for this reformulation and the same argument earlier regarding the decline of acceptances.

Significant as the results may see, the danger that the algorithm gets stuck in a local optimum exists. We speculate that it is due to the tendency to choose feasible neighbors of currently accepted feasible schedule as future solutions in the course of the algorithm. Also the interplay of search in both feasible and infeasible regions suffers a great downfall.

Also we observe that the algorithm terminates prematurely because the temperature value is rather high by the time the total number of rejections has been reached. This also plays a role in the "underexploitation" of the solution space.

| **Run #** | 4 | 5 | 8 |
|---|---|---|---|
| **Date** | 3/4/2006 | 3/4/2006 | 3/4/2006 |
| **Time** | 12:20:30 PM | 12:24:38 PM | 12:41:17 PM |
| | | | |
| **PARAMETERS** | | | |
| **Number of teams** | 6 | 6 | 6 |
| **Maximum iterations** | | | |
| **Initial temperature** | 100 | 100 | 200 |
| **Iterations per temperature** | | | |
| **Reduction factor** | 0.9 | 0.95 | 0.95 |
| **Acceptances** | | | |
| **Rejections** | 100 | 100 | 200 |
| **Deteriorations** | 10 | 10 | 20 |
| **Minimum temperature** | | | |
| **Weight of penalty** | 2000 | 2000 | 2000 |
| | | | |
| **RESULTS** | | | |
| **Initial cost** | 27868.000 | 27868.000 | 33260.000 |
| **Minimum cost** | 26240.330 | 26640.181 | 27545.010 |
| **Maximum cost** | 35717.053 | 36492.898 | 36159.027 |
| **Average cost** | 31891.464 | 30919.224 | 32232.905 |
| **Standard deviation** | 1881.073 | 1652.277 | 1714.415 |
| **Number of iterations** | 151 | 143 | 282 |
| **Number of acceptances** | | 1 | 17 |
| **Number of rejections** | 100 | 100 | 200 |
| **Number of deteriorations** | 51 | 42 | 65 |
| **Final temperature** | 59.049 | 81.451 | 171.475 |
| **Time (in seconds)** | 59.406 | 64.906 | 116.266 |
| | | | |
| **Accepted feasible schedules** | 51 | 43 | 82 |
| | | | |
| **Cost of best feasible schedule** | 29205.000 | 27868.000 | 28870.000 |
| **Time found (in seconds)** | 118 | 1.063 | 113.250 |
| **Iteration #** | 48.969 | 1 | 275 |

**Table 4.4:** SA performance of the reformulation

# Chapter 5

# Summary and Recommendations

In this chapter, we summarize the discussion that we had in the past chapters. Afterwards, some recommendations are given to further enhance the algorithm and at the same time provide some ideas for future works involving sports scheduling.

## 5.1 Summary

Arguably the most crucial activity in operating any sports leagues is the scheduling of games. However, creating schedules proved to be a difficult process especially when there are many teams and lots of requirements and preferences taken into consideration. A problem class was proposed wherein only the distance and flow issues are considered in creating a schedule for t teams, $t \geq 2$ and is even. This is known as the traveling tournament problem. The objective in TTP is to find the optimal schedule (in terms of cost), which is double-round robin and may or may not satisfy the atmost and norepeat soft constraints.

TTP is considered as a NP-hard problem. Several methods have been studied in order to generate near-optimal solutions to this problem. Among them is the Simulated Annealing metaheuristic. SA is an iterative, probabilistic algorithm whose methodology is patterned after the physical annealing process. Starting from an initial solution and given an initial temperature value, we obtain the neighbor of this solution by randomly selecting a neighborhood move. The corresponding cost of the obtained schedule is compared to the accepted cost so far. It is either accepted or rejected, based on a probability measure. The temperature is gradually lowered when a condition is satisfied involving certain predefined parameters and the search for the best possible solution resumes. Finally the algorithm is terminated when any parameter reached a preset value. The solution with the minimum cost, for any iteration, is selected as the optimal solution.

For our specific problem, the NL6 instance is considered. The formulation for this problem was presented. Using Visual Basic® for Excel, a program was created in order to implement SA to the particular problem selected. Several runs to this program were done and for each run the parameter values were varied. We observed that the optimal solution in most of the runs were not feasible. Hence, certain modifications were made to the program, primarily to the objective function.

## 5.2 Recommendations

Due to time constraints, we only focused on a particular TTP instance in our study. However, the Visual Basic® program which was developed can easily be adjusted and recoded to accommodate larger TTP instances.

Also we can consider the scheduling problems wherein there are additional soft constraints. After all, this is the case in real life tournaments. For example, a team must have equal weekend and weekday games or a team must not play its final game at its opponent's home court for two consecutive years. [GN], [RC] and [WCJ] are examples of studies that explore additional soft constraints.

Also, we can explore non-round robin leagues that can be broken down into phases that resemble TTP [MT3]. This is the case for many professional sports leagues, like the NBA, NFL, among others.

In this study, an attempt was made to vary the objective function. Some of the SA runs that were executed made use of the objective function wherein the goal is to minimize the penalty of a schedule. Although this experiment did not significantly improve the best feasible solution found, this can be a launching point for applying multiobjective programming to the TTP. Minimization of penalty (in order to improve feasibility of schedule obtained) and minimization of distance can be combined in order to find better solutions to various TTP instances. Also we can consider the use of other objective functions. For example, in [MT4], Michael Trick stated that the maximization of attendance or TV ratings is an unexploited concept in real sports leagues. Likewise, we can vary the penalty function used in the problem.

In [WCJ], the issue of fairness in the schedule was raised. It is said that in any game both teams get their share of the gate receipts, with the home team getting the majority. Both constraints and objective function can be formulated to focus on this important issue.

There are around 10 parameters in our SA algorithm. Most of these have a continuous range of values. Hence, there are infinitely many ways that we can run the SA algorithm presented in this paper. Also we can vary the initial solutions in each run. Furthermore, we can improve the SA by using hybrid algorithms (those methods that combine SA features with that of other metaheuristics). Finally, we can compare results with that of commercially released scheduling and other OR-related softwares.

# List of References

[AA] A. ANAGNOSTOPOULOS, ET.AL, *A Simulated Annealing Approach to the Traveling Tournament Problem*, http://tux.cs.brown.edu/pvh/cpaior03.pdf.

[ASc] A. SCHAERF, *Scheduling Sports Tournaments using Constraint Logic Programming*. Available at http://www.diegm.uniud.it/satt/papers/Scho99.pdf.

[ASu] A. SUZUKA, ET.AL, *Solving Sports Scheduling Problems using Network Structure*. Available at http://www.sk.tsukuba.ac.jp/~yoshise/Reports/sspgf.pdf.

[CR1] C. RIBEIRO AND S. URRUTIA, *Heuristics for the Mirrored Traveling Tournament Problem*. Available at http://www.esportemax.org/patat.ppt.

[CR2] _____, *OR on the Ball: Applications in sports scheduling and management*. Available at http://www.lionhrtpub.com/orms/orms-6-04/sports.html.

[EB] "Computer Science," *The New Encyclopaedia Britannica*, vol. 16, p. 635, 1989.

[FB] F. BIAJOLI, ET.AL, *Scheduling the Brazilian Soccer Championship: A Simulated Annealing Approach*. Available at http://www.optline.com.br/bssp.html.

[FH] F. HILIER AND G. LIEBERMAN, *Introduction to Operations Research 8th edition*, McGraw-Hill Higher Education, Boston, 2005.

[GN] G. NEMHAUSER AND M. TRICK, *Scheduling a Major College Basketball Conference*. Available at http://mat.gsia.cmu.edu/trick/acc.pdf.

[GTL] GAN TIAW LEONG, *Constraint Programming for the Traveling Tournament Problem*. Available at http://www.comp.nus.edu.sg/~henz/students/gan_tiaw_leong.pdf.

[HC] H. CRAUWELS AND D. VAN OUDHEUSDEN, *A Generate-and-Test Heuristic Inspired by Ant Colony Optimization for the Traveling Tournament Problem*. Available at http://ingenieur.kahosl.be/vakgroep/IT/Patat2002/SportTT/Crauwels.pdf.

[JB] J. BIRGE, *Scheduling a Professional Sports League in Microsoft® Excel: Showing Students the Value of Good Modeling and Solution Techniques*. Available at http://ite.pubs.informs.org/Vol5No1/Birge/Birge.pdf.

[JH] J. HAMIEZ AND J. HAO, *Solving the Sports League Scheduling Problem using Tabu Search*. Available at http://www.info.univ-angeri.fr/pub/hao/papers/ECAI00WS.pdf.

[JL] J. LEONARD, *Interactive Game Scheduling with Genetic Algorithm*. Available at http://goanna.cs.rmit.edu.au/~rc/papers/leonard-mbc.pdf.

[JP] J. PIRA, *School Timetabling by Genetic Algorithm and Simulated Annealing*, MS Thesis, University of the Philippines, 1999.

[KE] K. EASTON, ET.AL, *The Traveling Tournament Problem Description and Benchmarks*. Available at http://mat.gsia.cmu.edu/trick/ttp.pdf.

[KV] K. VALLESTEROS, *Simulated Annealing and Genetic Algorithm Applied to a Vehicle Routing Problem*, MS Thesis, University of the Philippines, 2003.

[LD] L. DIGASPERO AND A. SCHAERF, *A Tabu Search Approach to the TTP*. Available at http://ing.unite.it/eventi/reru05/articoli/DiGasperoSchaerf.pdf.

[LM] L. MORALES AND F. MALDONADO, *Constructing Optimal Schedules for Certain Tournaments using Tabu Search*. Available at http://www.dc.uba.ar/alio/io/pdf/1998/paper-10.pdf.

[MA] M. ADRIAEN, ET.AL, *An Agent Based Metaheuristic for the Traveling Tournament Problem*. Available at http://webhost.ua.ac.be/eume/workshops/reallife/adriaen.pdf.

[ME] *Microsoft® Excel/Visual Basic® Reference Second Edition*, Microsoft Press, Redmond, WA, 1995.

[MH] M. HENZ, *Scheduling a Major College Basketball Conference – Revisited*. Available at http://comp.nus.edu.sg/~henz/publications/ps/acc.ps.

[MT1] M. TRICK, *Adventures in Sports Scheduling*. Available at http://www.cs.cmu.edu/afs/cs.cmu.edu/project/aco/www/dimacs/trick.html.

[MT2] _____, *Challenge Traveling Tournament Instances*, Available at http://mat.gsia.cmu.edu/TOURN.

[MT3] _____, *Integer and Constraint Programming Approaches for Round Robin Tournament Scheduling.* Available at http://mat.gsia.cmu.edu/trick/tourn_final.pdf.

[MT4] _____, *Using Sports Scheduling to Teach Integer Programming*. Available at http://ite.pubs.informs.org/Vol5No1/Trick/Trick.pdf.

[PVH] P. VAN HENTENRYCK AND Y. VERGADOS, *Minimizing Breaks in Sport Scheduling with Local Search*. Available at http://www.cs.brown.edu/people/ pvh/breakmin.pdf.

[RC] R. CAMPBELL AND D. CHEN, “A Minimum Distance Basketball Scheduling Problem,” In R. Machol, et.al (eds.), *Management Science in Sports*, pp. 15-25, North-Holland Publishing Company, Amsterdam, 1976.

[TB] T. BENOIST, ET.AL, *Lagrangian Relaxation and Constraint Programming Collaborative Schemes for TTP.* Available at http://www.icparc.ic.ac.uk/~mgw/cpaior2001/paper29.final.doc.

[VB] *Visual Basic® User's Guide*, Microsoft Corporation, 1993-1994.

[WCJ] W. CAIN, JR., “The Computer-Assisted Heuristic Approach used to Schedule the Major League Baseball Clubs,” In R. Machol and S. Ladany (eds.), *Optimal Strategies in Sports*, pp. 32-41, North-Holland Publishing Company, Amsterdam, 1977.