Population-Based Simulated Annealing for Traveling Tournaments

Pascal Van Hentenryck and Yannis Vergados
Brown University, Box 1910, Providence, RI 02912

Abstract
This paper reconsiders the travelling tournament problem, a complex sport-scheduling application which has attracted significant interest recently. It proposes a population-based simulated annealing algorithm with both intensification and diversification. The algorithm is organized as a series of simulated annealing waves, each wave being followed by a macro-intensification. The diversification is obtained through the concept of elite runs that opportunistically survive waves. A parallel implementation of the algorithm on a cluster of workstations exhibits remarkable results. It improves the best known solutions on all considered benchmarks, sometimes reduces the optimality gap by almost 50%, and produces novel best solutions on instances that had been stable for several years.

Introduction
Sport scheduling has become a steady source of challenging applications for combinatorial optimization. These problems typically feature complex combinatorial structures (e.g., round-robin tournaments with side constraints), as well as objective functions measuring the overall quality of the schedule (e.g., travel distance).

The travelling tournament problem (TTP) is an abstraction of Major League Baseball (MLB) proposed by Easton, Nemhauser, and Trick 2001 to stimulate research in sport scheduling. The TTP consists of finding a double round-robin schedule satisfying constraints on the home/away patterns of the teams and minimizing the total travel distance. The TTP has been tackled by numerous approaches, including constraint and integer programming (and their hybridizations) (Easton, Nemhauser, & Trick 2001), Lagrangian relaxation (Benost, Laburthe, & Rottembouer 2001), and meta-heuristics such as simulated annealing (Anagnostopoulos et al. 2003; 2006), tabu search (Di Gaspero & Schaefer 2006), GRASP (Ribeiro & Urrutia 2007), and hybrid algorithms (Lim, Rodrigues, & Zhang 2006) to name only a few. The best solutions so far have been obtained by meta-heuristics, often using variations of the neighborhood proposed in (Anagnostopoulos et al. 2003). This includes new variations of the TTP based on circular and NFL distances. It is also interesting to observe that progress seems to have slowed down in recent years and some of the early instances have not been improved for several years.

This paper presents a population-based simulated annealing algorithm with both intensification and diversification components. The core of the algorithm is organized as a series of waves, each wave consisting of a collection of simulated annealing runs. At the end of each wave, an intensification takes place: a majority of the simulated annealing runs are restarted from the best found solution. Diversification is achieved through the concept of elite runs, a generalization of the concept of elite solutions. More precisely, at the end of each wave, the simulated annealing runs that produced the k-best solutions so far continue their execution opportunistically from their current states. This core procedure is terminated when a number of successive waves fail to produce an improvement and is then restarted at a lower temperature.

This population-based simulated annealing was implemented on a cluster of workstations. It produces new best solutions on all TTP instances considered, including the larger MLB instances which had not been improved for several years, the circular instances, and the NFL instances for up to 26 teams. Although simulated annealing is not the most appropriate algorithm for the circular instances, the population-based algorithm has improved all best solutions for 14 teams or more on these instances (and MLB 12). The improvements are often significant, reducing the optimality gap by almost 40% on many instances. The parallel implementation also obtained these results in relatively short times compared to the simulated annealing algorithm in (Van Hentenryck & Vergados 2006).

The broader contributions of the paper are twofold. First, it demonstrates the potential complementarity between the macro-intensification and macro-diversification typically found in tabu-search and population-based algorithms and the micro-intensification (temperature decrease) and micro-diversification (threshold acceptance of degrading move) featured in simulated annealing. Second, the pa-
per indicates the potential benefit of population-based approaches for simulated annealing, contrasting with recent negative results in (Onbaşoğlu & Özdamar 2001). The rest of the paper starts by reviewing the TTP and the simulated algorithm TTSA. It then presents the populated-based simulated annealing scheme and gives the experimental results on the TTP. The paper concludes by discussing related work.

Problem Description

A TTP input consists of $n$ teams ($n$ even) and an $n \times n$ symmetric matrix $d$, such that $d_{ij}$ represents the distance between the homes of teams $T_i$ and $T_j$. A solution is a schedule in which each team plays with each other twice, once in each team’s home. Such a schedule is called a double round-robin tournament. It should be clear that a double round-robin tournament has $2n - 2$ rounds. For a given schedule $S$, the cost of a team as the total distance that it has to travel starting from its home, playing the scheduled games in $S$, and returning back home. The cost of a solution is defined as the sum of the cost of every team. The goal is to find a schedule with minimal travel distance satisfying the following two constraints:

1. **Atmost Constraints**: No more than three consecutive home or away games are allowed for any team.

2. **Norepeat Constraints**: A game of $T_i$ at $T_j$’s home cannot be followed by a game of $T_j$ at $T_i$’s home.

As a consequence, a double round-robin schedule is feasible if it satisfies the atmost and norepeat constraints and is infeasible otherwise.

The Simulated Annealing Algorithm

This paper leverages the simulated annealing algorithm TTSA (Anagnostopoulos et al. 2006) which is used as a black-box. It is not necessary to understand its specifics which are described elsewhere. It suffices to say that TTSA explores a large neighborhood whose moves swap the complete/partial schedules of two rounds or two teams, or flip the home/away patterns of a game. The objective function $f$ combines the total distance and the violations of the atmost and norepeat constraints. TTSA uses strategic oscillation to balance the time spent in the feasible and infeasible regions.

Populated-Based Simulated Annealing

The core of the population-based simulated annealing receives a configuration $S$ (e.g., schedule) and a temperature $T$. It executes a series of waves, each of which consisting of $n$ executions of the underlying simulated annealing algorithm (in this case, TTSA). The first wave simply executes $SA(S,T)$ $N$ times (where $N$ is the size of the population). Subsequent waves consider both opportunistic and intensified executions. The simulated annealing runs that produced the $k$-best solutions so far continue their executions: hopefully they will produce new improvements and they provide the macro-diversification of the algorithm. The $N - k$ remaining runs are restarted from the best solution found so far and the temperature $T$.

Figure 1 illustrates the core of the algorithm for a population of size $N = 20$ and $k = 4$. Figure 1(a) shows that all executions starts from the same configuration and Figure 1(b) depicts the behavior during wave 1. The best solution obtained is $S_{8,1}$ (solid square in the figure). Several other executions also produces solutions ($S_{2,1}^*, S_{5,1}^*, S_{10,1}^*, S_{13,1}^*, S_{18,1}^*$) that improve upon their starting points (circles in the figure). The best 3 of them (solid circle), together with the best solution found so far, defines the elite runs used for diversification (i.e., $S_{2,1}^*, S_{8,1}^*, S_{13,1}^*, S_{18,1}^*$). Figure 1(c) depicts the start of the second wave. It highlights that the elite runs continue their execution from their current state, while the remaining 16 executions restarts from the best solution $S_{8,1}^*$ and the initial temperature. Figure 1(d) shows the executions of the first two waves. The second wave found a new best solution $S_{13,2}^*$ (produced by one of the elite runs), while several executions improve upon their starting points. The 4 best solutions are now $S_{2,2}^*, S_{8,1}^*, S_{13,2}^*$, and $S_{15,2}^*$, and the simulated annealing executions that produced them are now the set of elite runs. Figure 1(e) depicts the launch of the third wave. Observe that the two elite runs (those that produced $S_{8,1}^*$ and $S_{13,2}^*$) will now execute for the third successive wave, while two new ones have emerged.

This core procedure terminates after a number of stable waves, i.e., successive waves that have not improved the best solution. It is embedded in an outermost loop that progressively decreases the temperature $T$.

This overall algorithm is depicted in Figures 2 and 3. Figure 2 describes the core procedure PBSA-P for a population $P$ of size $N = |P|$. For each member $p$ of the population, the algorithm maintains its current starting configuration $S_p$ and temperature $T_p$, as well as the value $f_p$ of the best solution $p$ has generated. These variables are initialized in lines 2–6. The algorithm also maintains the best overall solution $S^*$ and the number stable of successive waves without improvement to $S^*$. Lines 8–23 are concerned with the execution of a wave. For each $p \in P$, PBSA-P applies the simulated annealing algorithm for $t$ units of time on configuration $S_p$ with starting temperature $S_p$. The simulated annealing execution returns the best solution $S_p^*$ of this run and the final configuration $S_p^+$ and temperature $T_p^+$ (line 8). If the run improves its starting solution, i.e., $f(S_p) < f(S_p^*)$, PBSA-P updates variable $f_p$ (line 11). If these runs have not improved the best solutions, the next wave continues each of the runs from their current state (see the instructions in lines 12–13 that implement this behavior). Otherwise, the runs that produced the $k$-best solutions (the elite runs) continue their executions, while the remaining $N - k$ runs (the set $R$ in line 19) are restarted from their current best solution.
Figure 1: Illustrating PBSA with $k = 4$.

Figure 2: PBSA-P: A Phase of PBSA

Figure 3: The Population-Based Simulated Annealing Algorithm PBSA

Experimental Results

Experimental Setting The algorithm was implemented in parallel to execute each run in a wave concurrently. The experiments were carried out on a cluster of 60 Intel-based, dual-core, dual-processor Dell Poweredge 1855 blade servers. Each server has 8GB of memory and a 300G local disk. Scheduling on the cluster is performed via the Sun Grid Engine, version
PBSA from High-Quality Solutions  The experimental results are summarized in Tables 1, 2, and 3, which report both on solution quality and execution times. With respect to solution quality, the tables describe the previous best solution (best) (not found by PBSA), the best lower bound (LB), the minimum (min) and average (mean) travel distances found by PBSA, and the improvement in the optimality gap (best - LB) in percentage (%G). The results on execution times report the times (in seconds) taken for the best run (Time(Best)), the average times (mean(T)), and the standard deviation (std(T)).

As far as solution quality is concerned, PBSA improves on all the best-known solutions for the MLB and circular instances with 14 teams or more. These MLB instances had not been improved for several years despite new algorithmic developments and approaches. PBSA also matches the best-known solution for 12 teams which the simulated annealing algorithm TTSA has not been able to obtain. It also improves the NFL instances for 16 to 26 teams (larger instances were not considered for lack of time). The improvement in the optimality gap is often substantial. For MLB-16, CIRC-20, and NFL-20, the improvements are respectively about 27%, 45%, and 48%.

As far as solution times are concerned, PBSA typically finds its best solutions in times significantly shorter than TTSA. On the MLB instances, PBSA found its new best solutions within 10 minutes, although these instances had not been improved for a long time. Typically, the new best solutions are found within an hour for problems with less than 20 teams and in less than two hours otherwise. These results are quite interesting as they exploit modern architectures to find the best solutions in competitive times, the elapsed times being significantly shorter than TTSA.

It is also interesting to look at PBSA's behavior. Figure 4 depicts the evolution of its objective function over time for MLB-16 and is representative of the typical behavior of the algorithm. The execution exhibits alternating sequences of fast improving and stagnant periods. Intuitively, repeated intensifications and diversifications allow the search to escape from very "difficult" local minima and to start improving again rapidly.

PBSA from Scratch  Figures 4 and 5 describe the performance of PBSA when the TTSA is only run shortly to produce a starting point. These results are particularly interesting. PBSA improves the best
known solutions for the MLB instances for 12, 14, and 16 teams and for the circular instances 12 and 14 (i.e., all the tested instances from scratch). For the MLB, the improvement for 14 teams is the same as when PBSA starts from a high-quality solution, the improvement for 16 teams is even better, and the improvement for 12 teams is new, reducing the optimality gap by about 14%, 7%, and 31% respectively. For the circular instances, the improvement is better than when PBSA starts from a high-quality solution, but not as good for 14 teams. The elapsed times are more significant but are still lower than the times for TTSA.

Related Work

This section briefly reviews the relationships between PBSA and cooperative parallel search, scatter search, and memetic algorithms.

Cooperative Parallel Search Population-based simulated annealing can be viewed as a cooperative parallel search. There is a great variety of proposed schemes of this kind, a large number of which based on simulated annealing. Onbaşoğlu et. al. 2001 provide an extensive survey of parallel simulated annealing algorithms and compare them experimentally on global optimization problems. They also classify those schemes into application-dependent and application-independent parallelization.1

In the first category, the problem instance is divided among several processors, which communicate only to deal with dependencies. For instance, in VLSI design, the processors specialize on different areas of the circuit. See (Greening 1990) for a detailed account of parallel simulated annealing techniques for VLSI design. In the second category, Onbaşoğlu et. al. further distinguish between asynchronous parallelization with no processor communication, synchronous parallelization with different levels of communication, and highly-coupled synchronization in which neighborhood solutions are generated and evaluated in parallel. In the first two cases, processors work on separate Markov chains while, in the third case, they cooperate on a single Markov chain.

Communication patterns between processors can take the form of simple transmission of cost values, occasional exchange of (possibly partial) solutions, or even intensive exchanges of solutions. Hybrid schemes combining different forms of communication have also been developed (e.g., (Kliewer & Tschöke 2000)). There are schemes that cannot be easily classified, such as the parallel simulated annealing in (Chu, Deng, & Reinitz 1999), in which the processors work on highly inter-dependent Markov chains by mixing states.

PBSA can thus be viewed as an application-independent algorithm with synchronous parallelization and periodic exchange of solutions. The scheme proposed in (Janaki Ram, Sreenivas, & Ganapathy Subramanian 1996) (which only exchanges partial solutions) and the SOEB-F algorithm (Onbaşoğlu & Özdamar 2001) are probably the closest to PBSA but they do not use diversification and elite runs. Observe also that SOEB-F typically fails to produce sufficiently good solutions (Onbaşoğlu & Özdamar 2001).

Communication patterns between processors can take the form of simple transmission of cost values, occasional exchange of (possibly partial) solutions, or even intensive exchanges of solutions. Hybrid schemes combining different forms of communication have also been developed (e.g., (Kliewer & Tschöke 2000)). There are schemes that cannot be easily classified, such as the parallel simulated annealing in (Chu, Deng, & Reinitz 1999), in which the processors work on highly inter-dependent Markov chains by mixing states.

PBSA can thus be viewed as an application-independent algorithm with synchronous parallelization and periodic exchange of solutions. The scheme proposed in (Janaki Ram, Sreenivas, & Ganapathy Subramanian 1996) (which only exchanges partial solutions) and the SOEB-F algorithm (Onbaşoğlu & Özdamar 2001) are probably the closest to PBSA but they do not use diversification and elite runs. Observe also that SOEB-F typically fails to produce sufficiently good solutions (Onbaşoğlu & Özdamar 2001).

It is also useful to point out that the above classification is not limited to simulated annealing. A cooperative parallel scheme based on tabu search is presented in (Asahiro, Ishibashi, & Yamashita 2003) and is applied to the generalized assignment problem. In this context, we can point out PBSA could be lifted into a

1There are also approaches attempting to parallelize sequential versions of simulated annealing (e.g., (Sohn 1995))

Table 5: Quality and Times in Seconds of PBSA for Circular Distances Starting From Scratch.

<table>
<thead>
<tr>
<th>n</th>
<th>Best</th>
<th>LB</th>
<th>min</th>
<th>mean</th>
<th>%G</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>408</td>
<td>384</td>
<td>404</td>
<td>418.2</td>
<td>16.6</td>
</tr>
<tr>
<td>14</td>
<td>654</td>
<td>590</td>
<td>640</td>
<td>654.8</td>
<td>21.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n</th>
<th>Time(Best)</th>
<th>mean(T)</th>
<th>std(T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>2200</td>
<td>1102.0</td>
<td>816.73</td>
</tr>
<tr>
<td>14</td>
<td>1720</td>
<td>1396.0</td>
<td>457.10</td>
</tr>
</tbody>
</table>
generic algorithm providing macro-intensification and macro-diversification for any meta-heuristic. Whether such a generic algorithm would be useful in other contexts remain to be seen however.

Memetic Algorithms and Scatter Search PBSA can be seen as a degenerated form of scatter search (Laguna & Marti 2003) where solutions are not combined but only intensified. Moreover, the concept of elite solutions is replaced by the concepts of elite runs which maintains the state of the local search procedures. PBSA can also be viewed as a degenerated form of memetic algorithms (Norman & Moscato 1991), where there is no mutation of solutions: existing solutions are either replaced by the best solution found so far or are “preserved”. Once again, PBSA does more than preserving the solution: it also maintains the state of the underlying local search through elite runs. It is obviously an interesting research direction to study how to enhance PBSA into an authentic scatter search and memetic algorithm. The diversification so-obtained may further improve the results.

Conclusions

This paper proposed a population-based simulated annealing algorithm for the TTP with both macro-intensification and macro-diversification. The algorithm is organized as a series of waves consisting of many simulated annealing runs. Each wave being followed by a macro-intensification restarting most of the runs from the best found solution and a macro-diversification which lets elite runs the chance to produce new best solutions. The algorithm was implemented on a cluster of workstations and exhibits remarkable results. It improves the best known solutions on all considered benchmarks, may reduce the optimality gap by almost 50%, and produces better solutions on instances that had been stable for several years. Moreover, these improvements were obtained by the parallel implementation in times that are significantly shorter than the simulated annealing algorithm TTSA, probably the best algorithm overall to produce high-quality solutions.

These results shed some light on the complementarity between the micro-intensification and micro-diversification inherent to simulated annealing and the macro-intensification and macro-diversification typically found in other meta-heuristics or frameworks. They also contrast with some recent negative results in parallel simulated annealing. Future research will be devoted to understand how to combine TTP solutions to produce scatter search and memetic algorithms for the TTP. These techniques have the benefits of producing structural diversification, which may be fundamental in improving the TPP results further.

References


