A Beam Search Approach to the Traveling Tournament Problem

Nikolaus Frohner, Bernhard Neumann, and Günther R. Raidl

Institute of Logic and Computation, TU Wien, Vienna, Austria
{nfrohner|raidl}@ac.tuwien.ac.at, e1634034@student.tuwien.ac.at

Abstract. The well-known traveling tournament problem is a hard optimization problem in which a double round robin sports league schedule has to be constructed while minimizing the total travel distance over all teams. The teams start and end their tours at their home venues, are only allowed to play a certain maximum number of games in a row at home or away, and must not play against each other in two consecutive rounds. The latter aspects introduce also a difficult feasibility aspect. In this work, we study a beam search approach based on a recursive state space formulation. We compare different state ordering heuristics for the beam search based on lower bounds derived by means of decision diagrams. Furthermore, we introduce a randomized beam search variant that adds Gaussian noise to the heuristic value of a node for diversifying the search in order to enable a simple yet effective parallelization. In our computational study, we use randomly generated instances to compare and tune algorithmic parameters and present final results on the classical National League and circular benchmark instances. Results show that this purely construction-based method provides mostly better solutions than existing ant-colony optimization and tabu search algorithms and it comes close to the leading simulated annealing based approaches without using any local search. For two circular benchmark instances we found new best solutions for which the last improvement was twelve years ago. The presented state space formulation and lower bound techniques could also be beneficial for exact methods like A^* or DFS^* and may be used to guide the randomized construction in ACO or GRASP approaches.

Keywords: Traveling Tournament Problem · Beam Search · Decision Diagrams

1 Introduction

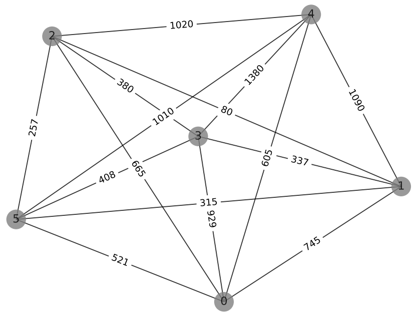
In 2001, Easton, Nemhauser, and Trick [4] introduced the traveling tournament problem (TTP). It concerns the construction of a double round robin schedule for a sports league, where the sum of the travel distances over all teams shall be minimized. Teams start and end at their respective home venues and are assumed to always travel directly from their current position to their next designated game venue, which is either at home or away. They are only allowed to play a certain maximum number of games away or at home consecutively, and two teams must not play against each other in two subsequent rounds. These aspects make even finding any feasible schedule in general difficult. At the time of writing, proven optimal solutions have been found for classical benchmarks instances with up to ten teams, but not for twelve and more teams, as stated on Michael Trick's TTP web page[1].

Due to the problem's complexity, many different metaheuristics have already been suggested to solve larger instances approximately. Neighborhood search based approaches as tabu search [3] or simulated annealing [1,14] provide particularly strong results. In this contribution, we present a beam search based on a new recursive state space formulation of the problem. We compare different lower bound heuristics to order the nodes in a layer of the state graph, which is traversed in breadth-first-search manner. Competitive results can be achieved with a randomized variant of the beam search in which we add noise to the heuristic estimates. This randomization enables a simple yet effective execution of multiple diversified beam search runs in parallel.

In Section 2 we summarize the work on which our new approach is based, specifically worth mentioning are the papers of Uthus, Riddle, and Guesgen [11,12,13], which broadly fall into the class of tree search based techniques. In particular, we build upon their bound pre-calculation method. We formally introduce the TTP in Section 3 and give an associated state space formulation in Section 4, which differs from integer programming or constraint programming formulations primarily used so far. A state may be reached by different partial schedules and determines the feasible completions to a complete schedule, which allows to detect and break symmetries on the go. Section 5 is concerned with the schedule construction algorithm on the state graph using beam search driven by lower bounds that are derived from the states. These lower bounds are calculated by solving either an associated traveling salesperson problem (TSP) or a capacitated vehicle routing problem (CVRP) independently for each team; the latter corresponds to the well-known independent lower bound (ILB) introduced by Easton et al [4].

We introduce a method to pre-calculate lower bounds for all states by means of decision diagrams. Moreover, we show in Section 6 how these bounds can be further tightened using the minimum number of trips (MNT) bound introduced by Urrutia et al. [10]. Section 7 presents computational results. We tune algorithmic parameters and compare the performance of different algorithm variants on randomly generated instances on a two-dimensional grid, and conduct final tests on the classical benchmark instances derived from teams of the US Major League Baseball (NL) and the circular instances (CIRC) [4]. We observe that our purely constructive approach, which does not make use of any local search, delivers competitive results. In particular, we could find new best solutions for two circular instances. Finally, we conclude in Section 8 and make suggestions for further research.

[1] `https://mat.tepper.cmu.edu/TOURN/`



$$\begin{pmatrix}
5 & -3 & 2 & 6 & -1 & -4 \\
-3 & 6 & 1 & 5 & -4 & -2 \\
-2 & 1 & -4 & 3 & -6 & 5 \\
-5 & 3 & -2 & -6 & 1 & 4 \\
4 & -5 & 6 & -1 & 2 & -3 \\
2 & -1 & 4 & -3 & 6 & -5 \\
6 & -4 & 5 & 2 & -3 & -1 \\
-4 & 5 & -6 & 1 & -2 & 3 \\
-6 & 4 & -5 & -2 & 3 & 1 \\
3 & -6 & -1 & -5 & 4 & 2
\end{pmatrix}$$

Fig. 1. Left: The NL6 problem instance from [4] shown as complete undirected graph. Right: A feasible double round robin tournament schedule represented by a $(2n-2)\times n$ matrix, where the value j of entry (r,i) corresponds to the game $i \to^r |j|$, if j is negative, otherwise to the game $j \to^r i$.

2 Previous Work

The TTP itself, together with the NL and CIRC benchmark instances, and the ILB were introduced by Easton et al. [4]. The MNT lower bound was proposed by Urrutia et al. [10] including an algorithm to calculate it. Uthus et al. [13] suggested an exact iterative deepening A* search, which allowed them to solve the NL instance with ten teams to proven optimality. Their approach features special symmetry breaking techniques, memoization, and was performed in parallel on 120 processors for a wall time of roughly 67 hours.

From this work, we adopt the method to pre-calculate independent lower bounds for states to occur during the state space traversal. We aim at solving larger instances approximately and compare our beam search results therefore to the results of today's state-of-the-art metaheuristic approaches, which are the simulated annealing from [1], the tabu search from [3], the ant colony optimization from [11], and the population-based simulated annealing from [14], where the latter found the so far best solutions for the larger NL and CIRC instances using a cluster consisting of 60 nodes.

For beam search in general, see, e.g., [6]. For a thorough introduction to decision diagrams in combinatorial optimization, we recommend the book by Bergman et al. [2].

3 Problem Formalization

We are given a set $V = \{1, \ldots, n\}$ of n teams, where n is even, and a distance matrix d where $d(i,j)$ is the traveling distance from team i's home venue to team j's home venue, $i, j \in V$. The goal is to find a double round robin tournament schedule, where every team plays at most U games subsequently at home or on the road (*at-most*), respectively, teams must not play against each other in subsequent rounds (*no-repeat*), and the total travel distance over all teams is to be minimized. Each team starts and ends at its home venue.

Adopting the formulation of [8], we see the teams V as vertices of a complete weighted directed graph $G = (V, A)$, where the weights are given by the distance matrix d. A double round robin schedule T is an ordered 1-factorization $T = (G^1 = (V, A^1), \dots, G^{2n-2} = (V, A^{2n-2}))$ of G, which is an ordered partitioning of the arcs into $2n-2$ perfect matchings (1-factors). An arc (i, j) (or $i \to^r j$) denotes that team i plays against team j at j's venue in round r, $r = 1, \dots, 2n-2$. The location of team i in round r is denoted $p_i^r \in V$ and determined by the single arc in A^r incident to team i. The objective value of a schedule T is the total travel distance given by

$$z(T) = \sum_{i=1}^{n} \left(d(i, p_i^1) + \sum_{r=2}^{2n-2} d(p_i^{r-1}, p_i^r) + d(p_i^{2n-2}, i) \right). \tag{1}$$

Throughout this paper and as in most previous work, we only consider $U = 3$, for which Thielen and Westphal [9] have shown strong NP-completeness in the corresponding decision variant of the problem.

Figure 1 shows on the left an example instance with $n = 6$ teams depicted as an undirected complete graph (distances are here assumed to be symmetric). A corresponding feasible TTP schedule is shown on the right, represented as a $(2n-2) = 10$ rounds by six teams matrix, denoting the opponent and venue for each round and team.

4 State Space Formulation

We model the solution space, i.e., the set of feasible schedules of a TTP instance (V, d), by a state graph. This is a rooted directed acyclic graph representing the feasible schedules by corresponding paths from a root state to a dedicated terminal state. The states (nodes) are organized into $n^2 - n + 2$ layers, where layer 0 only contains the root state s_r, layer $n^2 - n + 1$ only the terminal state s_t, and layers $l = 1, \dots, n^2 - n$ contain states representing the situations after the l-th played game.

Each state is a tuple $(\mathbf{M}^s, \mathbf{y}^s, \mathbf{r}^s, \mathbf{x}^s, \mathbf{h}^s, \mathbf{o}^s)$, where $\mathbf{M}^s = (M^s_{i,j})_{i,j \in V} \in \{0,1\}^{n \times n}$ is an incidence matrix that indicates the games left to be scheduled and vectors $\mathbf{y}^s = (y_i^s)_{i \in V}$, $\mathbf{r}^s = (r_i^s)_{i \in V}$, $\mathbf{x}^s = (x_i^s)_{i \in V}$, $\mathbf{h}^s = (h_i^s)_{i \in V}$, and $\mathbf{o}^s = (o_i^s)_{i \in V}$ represent for each team i the currently forbidden opponent y_i^s, the current round r_i^s, its location x_i^s, and the number of still possible home or away games left to play in a row h_i^s and o_i^s, respectively. The forbidden opponents $\mathbf{y}^s$ are used to implement the *no-repeat* constraint and $\mathbf{h}^s$ and $\mathbf{o}^s$ to take care of the *at-most* constraints. Moreover, this information contained in a state implies for each team $i \in V$ the set P_i^s of the games that can be played next without violating the TTP constraints.

A state transition from a state s at layer l to a state s' at layer $l+1$, $l = 0, \dots, n^2 - n$, corresponds to a specific game $i \to^r j$ being played by teams i and j at j's venue in round r. Each state transition is weighted by the sum

Partial schedule

$$\begin{pmatrix} 5 & -3 & 2 & 6 & -1 & -4 \\ -3 & 6 & 1 & 5 & -4 & -2 \\ -2 & 1 & -4 & 3 & \mathbf{-6} & \mathbf{5} \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \end{pmatrix}$$

$$\mathbf{M}^s = \begin{pmatrix} 0&0&0&1&1&1 \\ 1&0&0&1&1&1 \\ 1&1&0&0&1&1 \\ 1&1&1&0&1&1 \\ 0&1&1&0&0&1 \\ 1&0&1&0&1&0 \end{pmatrix} \rightarrow \mathbf{M}^{s'} = \begin{pmatrix} 0&0&0&1&1&1 \\ 1&0&0&1&1&1 \\ 1&1&0&0&1&1 \\ 1&1&1&0&1&1 \\ 0&1&1&0&0&0 \\ 1&0&1&0&1&0 \end{pmatrix}$$

$$\mathbf{x}^s = \begin{pmatrix} 1\\1\\3\\3\\3\\1 \end{pmatrix} \rightarrow \mathbf{x}^{s'} = \begin{pmatrix} 1\\1\\3\\3\\5\\5 \end{pmatrix} \quad \mathbf{o}^s = \begin{pmatrix} 1\\3\\2\\3\\1\\1 \end{pmatrix} \rightarrow \mathbf{o}^{s'} = \begin{pmatrix} 1\\3\\2\\3\\0\\3 \end{pmatrix} \quad \mathbf{h}^s = \begin{pmatrix} 3\\1\\3\\0\\3\\3 \end{pmatrix} \rightarrow \mathbf{h}^{s'} = \begin{pmatrix} 3\\1\\3\\0\\3\\2 \end{pmatrix}$$

Fig. 2. Left: Exemplary partial schedule for an instance with six teams before ending the third round, for which the teams six and five (in bold) are selected to play the next game. Right: Corresponding state updates where in the matrix of the games left the currently forbidden games implied by $\mathbf{y}^s, \mathbf{o}^s, \mathbf{h}^s$ are grayed out. We omitted $\mathbf{r}^s$ and $\mathbf{y}^s$ for space reasons.

of the distances both teams have to travel from their previous locations to play the game

$$\Delta z(s, s') = d(x_i^s, x_i^{s'}) + d(x_j^s, x_j^{s'}). \tag{2}$$

Teams for the game are selected in a way that the partial schedule grows round by round in ascending order where each round is completed before the next one starts. All paths starting from the root state and leading to the terminal state correspond to feasible solutions. Paths that end before the terminal state at a state without further transitions represent partial schedules that cannot be feasibly continued. A shortest path from the root to the terminal state therefore corresponds to an optimal feasible solution for a given problem instance.

We introduce two special rounds $r = 0$ and $r = 2n-1$ where every team is at its home location. Let $\mathbf{M}^{s_\mathrm{r}}$ be the matrix with non-diagonal ones and diagonal zeros, corresponding to all games to be played, and matrix $\mathbf{M}^{s_\mathrm{t}}$ be the all-zeros matrix. If there is no forbidden opponent for a team $i \in V$ in state s, then y_i^s is set to -1. The root state is then $s_\mathrm{r} = (M^{s_\mathrm{r}}, \mathbf{y}^{s_\mathrm{r}} = (-1, \ldots, -1), \mathbf{r}^{s_\mathrm{r}} = (0, \ldots, 0), \mathbf{x}^{s_\mathrm{r}} = (1, \ldots, n), \mathbf{h}^{s_\mathrm{r}} = (U, \ldots, U), \mathbf{o}^{s_r} = (U, \ldots, U))$ and the terminal state $s^\mathrm{t} = (\mathbf{M}^{s_\mathrm{t}}, \mathbf{y}^{s_\mathrm{t}} = (-1, \ldots, -1), \mathbf{r}^{s_\mathrm{t}} = (2n-1, \ldots, 2n-1), \mathbf{x}^{s_\mathrm{t}} = (1, \ldots, n), \mathbf{h}^{s_\mathrm{t}} = (0, \ldots, 0), \mathbf{o}^{s_\mathrm{t}} = (0, \ldots, 0))$. Transitions to the terminal state are special in the sense that they do not correspond to played games but just to going back to the teams' home venues.

Transitions from a state s at some layer l to a subsequent state s' at layer $l+1$ are done by selecting a game $(i, j) \in P_i^s$ (or (j, i)) where we impose the condition $r_i = r_j = \min_{i \in V} r_i$. This ensures that the teams are in the same round and games are assigned to teams round by round. If there exists a *dead* team i with $P_i^s = \emptyset$, our current state has no feasible completion. Since there is no meaning in which order we select the teams in a specific round r, we break this symmetry by defining a specific team permutation $\pi\colon V \to V$. At each state of layer l, a game from $P^s_{\pi_i}$ has to be played for which i and r_{π_i} are minimal. A trivial ordering is the lexicographic ordering of the teams.

Selecting the game (i, j) yields state s' with $\mathbf{M}^{s'}$ being a copy of $\mathbf{M}^s$ except that $M^{s'}_{i,j} = 0$, which implicitly removes this game from $P^{s'}_i$ and $P^{s'}_j$ as well. The position, round, and streak related information of i and j is updated from s to s' accordingly. To respect the *no-repeat* constraints, the forbidden opponent vector $\mathbf{y}^s$ is copied to $\mathbf{y}^{s'}$ except that $y^{s'}_i = j$ and $y^{s'}_j = i$, if $M^s_{j,i} = 1$; otherwise these values are set to -1. For every other team $k \in V \setminus \{i, j\}$, $y^{s'}_k = -1$ is set if $y^s_k \in \{i, j\}$. The *at-most* constraints are already implied by the updates in $\mathbf{o}^{\mathbf{s}'}$ and $\mathbf{h}^{\mathbf{s}'}$. If $o^{s'}_i = 0$, then away games are not allowed in the next round for team i; analogously, a continuation of j's home stand is not allowed, if $h^{s'}_j = 0$.

An exemplary state transition is shown in Fig. 2 for an instance with six teams before and after ending the third round with the game $(5, 6)$. We see that team five hits its away streak limit and all its away games are not available for the next round and that the game $(6, 5)$ is forbidden.

5 Beam Search

We perform a layer-by-layer breadth-first-search traversal of the state graph, where for each state all permitted games for a selected team are played by performing the respective transitions to corresponding successor states. The current shortest path value and the corresponding partial schedule are cached for each state during construction and updated if a shorter path to an already visited state is discovered.

Due to the complexity of the problem, only instances with four teams admit a complete construction of the state graph, providing a guaranteed optimal solution. We therefore restrict the search to an incomplete beam search where at each layer at most β states are kept for further consideration; parameter β is hereby called the beam width. In this way the total number of expanded states is polynomially bounded by $\mathcal{O}(n^2\beta)$. The shortest path through such a restricted state graph then corresponds to a feasible heuristic solution. To guide the search, in each layer the β most promising states are kept according to some state ranking heuristic, in the hope that the finally shortest path corresponds to an optimal or close-to-optimal solution. Classical beam search sorts the states by an f-value known from A* search that combines the length of the currently shortest path $g(s)$ to the state s with a lower bound $b(s)$ (or heuristic estimate) for the further continuation to the terminal state:

$$f(s) = g(s) + b(s) \tag{3}$$

In our beam search implementation, we only keep the current layer in a queue and the successive layer in a priority queue sorted according to f that contains at most the β best successor states so far. The latter is implemented by a maximum heap combined with a hash map to access arbitrary states in expected constant time. Before creating a successor state, we check in case of a full beam by means of incremental evaluation whether the potential new state's f-value is worse than the worst f-value in the heap. If this is the case, we do not need to consider the state further. Otherwise we create the successor state and check whether it already exists in the maximum heap, in which case we conditionally update its shortest path value and current best partial schedule. If the state was not yet contained in the heap, we replace its so far worst state by the newly created successor state. This approach gives us a smaller memory footprint than storing all created states until termination, allowing us to test higher beam widths. The current partial schedule is cached along each state in a growing vector.

As will be discussed in detail in Section 6, the lower bound values are also cached along the state for each team, together with the number of home and away games left for each team. The latter allow to quickly check whether there are not enough home games in relation to away games or vice versa to make a feasible completion.

To allow a simple parallelization of the beam search by independent diversified runs, we further introduce a randomized variant of the beam search. To this end we add a normally distributed random offset with standard deviation σ to each state's original f-value:

$$\tilde{f}(s) = f(s) + \mathcal{N}(0, \sigma). \tag{4}$$

The motivation is that states which would be pruned when just considering their deterministic f-value get a chance to survive, and they may possibly lead to superior solutions. Initially promising states can also get cut off early by drawing a too high random offset. Crucial is the standard deviation σ for which we make the following parameterized ansatz:

$$\sigma = \sigma_{\mathrm{rel}} \cdot b(s^{\mathrm{r}}) \tag{5}$$

Parameter σ_{rel} thus determines the fraction of the lower bound of the root state to be used as σ, so that the order of magnitude of the expected solution length of a given instance is respected. Tuning results for this parameter are presented in the computational study in Section 7.

Algorithm 1 shows our beam search in pseudo-code. next-team(l, s) selects the team to consider for a given layer l and state s. Trivial options are to take the lexicographically smallest team that is in a minimal round or to initially fix a random permutation of the teams.

Procedure feasibility-and-optimality-check$(H, \beta, s, b, \epsilon, (i, j))$ incrementally checks whether the transition would lead to a state for which we know for sure that it does not have a feasible completion. This is the case when not enough home or away games are available for a specific team to not violate the *at-most* constraint or because one team has an empty possible games set in this round. The optimality check is done by considering the increase in the f-value by the move and whether it is worse than the maximal f-value in a full, i.e., containing β states, maximum heap H—then the transition for game (i, j) does not need

Input: number of teams n, distance matrix d, root state s^{r}, terminal state s^{t}, noise parameter σ_{rel}, state lower bound function b, beam width β
Output: feasible schedule T

```
1  queue $Q \leftarrow \{s^{\mathrm{r}}\}$;
2  for $l \leftarrow 1$ to $n^2 - n$ do
3      $H \leftarrow$ empty maximum heap;
4      while $Q \neq \emptyset$ do
5          $s \leftarrow Q$.pop;
6          $t \leftarrow$ next-team$(l, s)$;
7          foreach $(i, j) \in \{(i', j') \in P_t^s | r_{i'}^s = r_{j'}^s\}$ do
8              $\epsilon \leftarrow \mathcal{N}(0, \sigma_{\mathrm{rel}} \cdot b(s^{\mathrm{r}}))$;
9              if feasibility-and-optimality-check$(H, \beta, s, b, \epsilon, (i, j))$ then
10                 $s' \leftarrow$ copy $s$ and make transition by playing $(i, j)$ and updating
                    state along with cached data accordingly;
11                 $s'.current\_schedule \leftarrow s'.current\_schedule \cup (i, j)$;
12                 $f(s') \leftarrow g(s') + b(s') + \epsilon$;
13                 include $s'$ into $H$ respecting $f(s')$;
14                 if $H.size > \beta$ then
15                     remove worst element of $H$;
16                 end
17             end
18         end
19     end
20     $Q \leftarrow$sorted-by-f-value$(H)$;
21 end
22 if $Q \neq \emptyset$ then
23     create going home transitions for all states $Q$ to $s^{\mathrm{t}}$;
24     return $s^{\mathrm{t}}.current\_schedule$;
25 else
26     return $\emptyset$;
```

Algorithm 1: Beam search for the TTP.

to be considered and state s' is not created, which saves a costly state expansion operation that would require to copy the whole current state and cache variables.

After all successor states have been checked and potentially included into the heap H, its states are transferred to queue Q, sorted according to state priorities, and thus these nodes become the new current layer. The sorting is done to fill the beam earlier with likely better states to increase the odds for rejecting the creation of successor states during incremental evaluation.

In the next section, we study in detail the crucial part of devising and efficiently calculating a lower bound b for determining the state's f-value.

6 Lower Bounds Calculation

The main idea for obtaining lower bounds is to relax the problem by considering the tours of all teams independently. Easton et al. [4] already suggested the independent lower bound (ILB) that applies this principle. This bound neglects the *no-repeat* constraints and considers only the away teams for a given team $i \in V$ with only the away *at-most* constraints. This amounts to a capacitated vehicle routing problem (CVRP), where the depot is at i's home venue, the customers are the away teams with unit demand, and the capacity for the trucks is $U = 3$. The CVRP itself is strongly NP-hard but for few customers tractable in practice.

Given an arbitrary state s and team i, we have to consider the remaining away teams $\mathcal{A}_i^s$ for i, the position x_i^s, and the remaining away streak o_i^s. If $x_i^s \neq i \wedge o_i^s = 0$, then we consider an artificial state in which the team is assumed to have returned home (this is the only option it has at that moment), $o_i^s = \min(\mathcal{A}_i^s, U)$, and add $d(x_i^s, i)$ to the resulting bound. Let the optimal total length for this problem for team i be $b_i^{\mathrm{CVRP}}(s)$. Then the sum of the optimal values over all teams is a lower bound for the optimal value of the corresponding TTP-feasible completion of s

$$b^{\mathrm{CVRP}}(s) = \sum_{i=1}^{n} b_i^{\mathrm{CVRP}}(s). \tag{6}$$

A natural further relaxation is to drop even the away *at-most* constraints, which yields a traveling salesperson problem based lower bound b^{TSP}. In this case, we do not have to consider the current away streak of team i in state s.

To provide better guidance for the beam search, we are more interested in tighter lower bounds, while keeping their computational costs in mind. A first natural strengthening is to consider also the home *at-most* constraints. Let $h_i^{\mathrm{left}} = |\mathcal{H}_i^s|$ be the number of home games left for team i in state s. Then we need at least $\tilde{h}_i^{\min} = \lceil (h_{\mathrm{left}} + \bar{h}_i)/U \rceil$ home stands to accommodate for the home games, where $\bar{h}_i$ is the length of the current home stand. Translated to the CVRP, this amounts to the constraint that we need to perform at least $\tilde{h}$ non-trivial tours. Analogously, every away streak needs at least one home game from where it came, minus one if the team is currently at home. This gives us a maximum to the home stands $\tilde{h}_i^{\max}$ we can realize from a given state. We can therefore define a home stand constrained lower bound $b_i^{\mathrm{CVRPC}}(s, \tilde{h})$ and tighten the CVRP bound by finding the minimum within the range of allowed home stands, summed over all teams resulting in the CVRP with home stands bound (CVRPH)

$$b^{\mathrm{CVRPH}}(s) = \sum_{i=1}^{n} \min_{\tilde{h} \in \{\tilde{h}_i^{\min}, \ldots, \tilde{h}_i^{\max}\}} b_i^{\mathrm{CVRPC}}(s, \tilde{h}). \tag{7}$$

To speed up our beam search, we pre-calculate the lower bounds for the states that can occur for a given TTP instance, similarly as done by Uthus et al. [13].

We do this by representing the whole space of feasible solutions to the given CVRP instance for each team i with an exact multi-valued decision diagram (DD) [2] and finally store the lower bounds for the states that occurred in a lookup table. Each node in this DD is associated with a state q consisting of the away games left to play $\mathcal{A}^q$ (represented by the subset of other teams against which team i still has to play), the team i's position x^q and the current number of consecutive away games, the away streak $\bar{o}^q$. The root state for a given team i is therefore $q_\mathrm{r} = (\{1,\ldots,i-1,i+1,\ldots,n\}, i, 0)$. Transitions are made until the terminal state $q_\mathrm{t} = (\{\}, i, 0)$ is reached, where in every layer, all available transitions are performed. Hereby we distinguish between three possibilities:

- Select any away team left $j \in \mathcal{A}^q$ to visit next, if there are such, and go home afterwards, where costs $d(x^q, j) + d(j, i)$ accrue. The state is updated accordingly to $(\mathcal{A}^q \setminus \{j\}, i, 0)$.
- If $\bar{o}^q$ is less than $U-1$, then select any away team left $j \in \mathcal{A}^q$ to visit next, if there are such, and stay at j afterwards, where costs $d(x^q, j)$ accrue. The state is updated to $(\mathcal{A}^q \setminus \{j\}, j, \bar{o}^q + 1)$.
- If $\mathcal{A}^q$ is empty, go home if not already at home, where costs $d(x^q, i)$ accrue. The state is updated to the terminal state q_t.

Paths from the root to the terminal node in the DD then correspond to the feasible solutions of the CVRP, and with the costs associated with the transitions (i.e., arcs in the DD), the lengths of such paths correspond to solution lengths. For each node in the CVRP decision diagram the shortest path to the terminal node is calculated and saved in the lookup table, serving as a lower bound for a team with given away teams to play, being at a position either at home or at some away team and its current away streak. Being a layered directed acyclic multigraph, the shortest paths for each node in the decision diagram can be calculated efficiently by doing a breadth-first-search backwards from the terminal to the root node.

The TSP based bound values can also be pre-calculated by these method by simply ignoring the away streak and allowing always a direct transition to a next away team without going home first.

Furthermore, for the CVRPH bound, we consider constrained shortest path lengths $z^{\mathrm{sp}}(q, \tilde{h})$ from any node to the terminal node, with the constraint that exactly $\tilde{h}$ home stands occur. This means that at most $\tilde{h}U - \bar{h}_i$ home games can be played from a given node, where $\bar{h}_i$ is the length of the current home stand for team i. At the terminal node $z^{\mathrm{sp}}(q_\mathrm{t}, 1) = 0$ and ∞ for each other node. In the backward sweep from q' to q with arc costs $c_{q,q'}$, if $x_q \neq i$, then we set $z^{\mathrm{sp}}(q, \tilde{h}) = \min\{z^{\mathrm{sp}}(q', \tilde{h}) + c_{q,q'}, z^{\mathrm{sp}}(q, \tilde{h})\}$. If on the other hand $x^q = i$, i.e., a new home stand has occurred, we set $z^{\mathrm{sp}}(q, \tilde{h}+1) = \min\{z^{\mathrm{sp}}(q', \tilde{h}) + c_{q,q'}, z^{\mathrm{sp}}(q, \tilde{h}+1)\}$. For each state we now have all the constrained lower bound values available that correspond to $b_i^{\mathrm{CVRPC}}(s, \tilde{h})$. Additionally, we define $b_i^{\mathrm{CVRPC},\geq}(s, \tilde{h}) = \min_{\tilde{h}' \in \{\tilde{h},\ldots,\tilde{h}_i^{\max}\}} b_i^{\mathrm{CVRPC}}(s, \tilde{h}')\ \forall \tilde{h} \in \{\tilde{h}_i^{\min},\ldots,\tilde{h}_i^{\max}\}$, which gives us the lower bounds when using *at least* $\tilde{h}$ home stands.

Table 1. Memory demand for different lower bound lookup tables over the number of teams in GB assuming two bytes per bound value.

n	TSP	CVRP	CVRPH
14	0.003	0.009	0.127
16	0.016	0.047	0.75
18	0.079	0.237	4.27
20	0.390	1.172	23.43

Table 2. Runtimes in minutes for CVRPH bound calculations for NL14 to NL16 and CIRC14 to CIRC18.

	14	16	18
NLn	25	169	-
CIRCn	25	173	903

In Table 1, we see the memory demand for the three different lower bound lookup tables in GB assuming 2 bytes per bound value. Up to 16 teams, they all have reasonable size and our experiments have shown that the 16 teams instance bounds can be pre-calculated within three hours in a prototypic Python 3.7 implementation on an Intel Xeon E5-2640 processor with 2.40 GHz in single-threaded mode, see Table 2. 18 teams instances are also within reach with the strong CVRPH bound taking already 15 hours—we suppose that an order of magnitude in time can be saved using a compiled language. For larger instances, these numbers and the computation times increase dramatically, since the number of bounds grows for the CVRPH bound with $\mathcal{O}(n^3 2^n)$—already the TSP bound, being the weakest, needs 42 GB for 26 teams.

For even a further tightening of the CVRPH bound, we make use of the minimum number of trips (MNT) bound by Urrutia et al. [10]. It does not assume strong independence between the teams anymore. Instead, the relaxed CONSTANT variant of the problem, where all distances are set to one is solved to optimality (or taking a lower bound), yielding a minimum number of trips all teams together have to perform in a feasible solution of the problem. A trip in this case is an atomic movement of a team from one venue to another. Given a state s, let us call $\tau = \sum_{i=1}^{n} t_i$ the number of trips performed so far by all teams in the shortest path from s_r to s. By the CVRPC bound, each team has an optimal number $\tilde{h}_i^\text{opt}$ of home stands from s to s_t. This translates to an optimal number of trips $t_i^\text{opt} = |\mathcal{A}_i^s| + \tilde{h}^\text{opt} - 1_{x_i=i}$. Let τ^lb be the lower bound for the minimum number of trips. If $\tau^\text{lb} \leq \tau + \sum_i t_i^\text{opt}$, then we cannot tighten the CVRPH bound further. Otherwise, we can add the constraint that the teams have to perform $\Delta\tau = \tau^{\mathrm{lb}} - \tau - \sum_i t_i^{\mathrm{opt}}$ extra trips, yielding the MNT bound

$$b^{\mathrm{MNT}}(s) = \min \quad \sum_{i=1}^{n} b_i^{\mathrm{CVRPC},\geq}(s, \tilde{h}_i) \tag{8}$$

$$\text{s.t.} \quad \tilde{h}_i \in \{\tilde{h}_i^{\min}, \ldots, \tilde{h}_i^{\max}\} \quad \forall i \in 1, \ldots, n \tag{9}$$

$$\tau + \underbrace{\sum_{i=1}^{n} |\mathcal{A}_i^s|}_{|P^s|} + \tilde{h}_i - 1_{x_i = i} \geq \tau^{\mathrm{lb}} \tag{10}$$

This bound can be calculated by solving a corresponding integer linear program using binary decision variables $y_i^{\tilde{h}}$ with costs derived from the CVRPC lower bound function $c_i^{\tilde{h}}$ and counting values $d_i^{\tilde{h}} \in \{\tilde{h}_i^{\min}, \ldots, \tilde{h}_i^{\max}\}$:

$$b^{\mathrm{MNT}}(s) = \min \quad \sum_{i=1}^{n} c_i^{\tilde{h}} y_i^{\tilde{h}} \tag{11}$$

$$\text{s.t.} \quad \sum_{\tilde{h}} y_i^{\tilde{h}} = 1 \quad \forall i \in 1, \ldots, n \tag{12}$$

$$\sum_{i,\tilde{h}} y_i^{\tilde{h}} d_i^{\tilde{h}} \geq \tau^{\mathrm{lb}} - |P^s| - \tau + \sum_{i=1}^{n} 1_{x_i = i} \tag{13}$$

We take the τ^{lb} values from [7], where Rasmussen and Trick present a Benders decomposition approach to solve the CONSTANT instances for up to 16 teams each within at most five minutes.

7 Computational Study

We conducted all our experiments on Intel Xeon E5-2640 processors with 2.40 GHz in single-threaded mode and a memory limit of 32GB. We implemented our approach as a prototype in Python 3.7, being aware that an implementation in a compiled language would likely be substantially faster and have a smaller memory footprint. To solve the integer linear programs for the MNT bound, we used Gurobi 12.8.

In Table 3, we present a comparison of the results obtained by the deterministic beam search with beam widths 1000 and 10000 for the instance sets NL and CIRC[2] [4] over the different lower bounds used in the state ordering. We see that the weak TSP bound does not provide good guidance and even misguides the search for larger instances, where using no bound ($b(s) = 0$) and sorting the nodes only by the currently shortest path length to them (SHORT) provides better results. Much better guidance can be observed for the CVRP based bounds, where we see similar improvements over all instances.

[2] `https://mat.tepper.cmu.edu/TOURN/`

Table 3. Final solution lengths of deterministic beam search with different state ordering heuristics and beam widths with lexicographic team orderings. Sorting the states by currently shortest path length to them (SHORT) does not use any lower bound; for a description of the TSP, CVRP, CVRPH, MNT lower bounds refer to Section 6. For circ16 with MNT we did not achieve a final result due to excessive runtime.

	$\beta = 1000$					$\beta = 10000$				
inst	SHORT	TSP	CVRP	CVRPH	MNT	SHORT	TSP	CVRP	CVRPH	MNT
nl6	24876	24759	23954	**23916**	**23916**	24876	**23978**	**23916**	**23916**	**23916**
nl8	42308	41977	40687	40687	40687	40970	41762	**39776**	**39776**	**39776**
nl10	67094	66469	62329	60713	62400	66087	64700	61129	60757	**60554**
nl12	131046	129209	116976	114499	114499	127238	119271	**113294**	114475	114824
nl14	217763	233765	211643	211116	211116	224537	219708	203519	**203279**	**203279**
nl16	309227	-	283985	285326	286085	301989	322567	276599	275562	**271251**
circ6	66	**64**	**64**	**64**	**64**	**64**	**64**	**64**	**64**	**64**
circ8	144	146	**134**	**134**	**134**	136	142	**134**	**134**	**134**
circ10	280	284	264	262	266	268	276	**246**	**246**	250
circ12	452	502	428	430	430	444	468	**418**	**418**	418
circ14	734	774	674	672	672	710	760	668	**656**	**656**
circ16	-	-	1012	1000	990	1012	1114	**956**	966	n/a

Since the number of classic benchmark instances is limited and to further validate the guidance quality of the different bounds, we created two different types of random instances. First, the set $\mathcal{I}_{L^1}$, where we sample 30 instances for team numbers 8, 10, and 12 each on an integer grid of size 1000×1000 using Manhattan distances to compute the resulting distance matrices; second, the set $\mathcal{I}_{L^2}$, using the same sampling procedure but using the rounded to the nearest even integer Euclidean distances. This yields in total 180 additional test instances. We exclude the TSP bound from our further experiments since it did not show promising results for the tests on the NL and CIRC instances. In Table 4, we see mean values of final solution lengths and corresponding standard deviations when performing the deterministic beam search with the different state ordering heuristics on the randomly generated instances. The gap between SHORT and CVRP is well observed especially with 10 and 12 teams. The gap between CVRP and CVRPH is closer, a Wilcoxon signed rank sum test shows that we can reject the assumption that CVRP is better than CVRPH with a significance level of $\alpha = 1\%$. The difference between CVRPH and MNT is for the L^1 distance instances inconclusive, and for the L^2 instances slightly in favor of MNT, but at the cost of substantially higher runtime due to the linear programs that need to be solved for every state. For further experiments we therefore limit ourselves to the CVRP/CVRPH bounds.

Finally, Table 5 compares our randomized beam search variant with either lexicographic or random team ordering performed in parallel and independently on 30 cores with several state-of-the-art approaches on three difficult NL and CIRC instances. Each beam search run was conducted with beam width $\beta = 10^5$

Table 4. Comparison of our beam search algorithm with $\beta = 1000$ over different state ordering heuristics on 180 randomly generated test instances with 8, 10, and 12 teams, using Manhattan and Euclidean distances, evenly split. Mean values of final solution lengths and standard deviations over 30 test instances are shown.

	$\beta = 1000$			
class	SHORT	CVRP	CVRPH	MNT
$\mathcal{I}^{8}_{L1}$	42532 ± 5384	40530 ± 5214	40405 ± 5030	40405 ± 5030
$\mathcal{I}^{10}_{L1}$	70049 ± 7280	65483 ± 6886	64760 ± 6689	64964 ± 6922
$\mathcal{I}^{12}_{L1}$	99086 ± 7991	92838 ± 8089	91728 ± 7726	91465 ± 7694
$\mathcal{I}^{8}_{L2}$	34412 ± 5088	33034 ± 5109	32965 ± 5071	32965 ± 5071
$\mathcal{I}^{10}_{L2}$	55019 ± 5872	51723 ± 5988	51269 ± 5808	51057 ± 5829
$\mathcal{I}^{12}_{L2}$	79699 ± 7293	74231 ± 6933	73700 ± 6456	73524 ± 6403

and randomization parameter $\sigma_{\text{rel}} = 0.001$ resulting in equally gentle noise applied to the f-values of the states in every layer. The noise parameter was determined using irace [5] on the randomly generated instances. The table shows minimum and mean values for solution lengths of finally best solutions. We observe that we can compete well with the other mainly constructive approach "ant colony optimization with forward checking and conflict-directed backjumping" (AFC-TTP) from [11] and the composite-neighborhood tabu search (CNTS) from [3] on the NL instances and obtain better results than these for the CIRC instances, without hybridizing with a final local search. For CIRC instances we can also obtain similar results to population-based simulated annealing from scratch (PBSAFS) from [14], which uses parallel simulated annealing. For the circular instances with 14 and 16 teams, we found new best feasible solutions, as of the time of writing according to Michael Trick's TTP web page. The strongest results overall for NL and CIRC are provided by simulated annealing (TTSA) from [1] and its parallel variant PBSA from [14].

Runtimes of our approach are shown in Fig. 3 measured for deterministic beam search on the NL instances up to 16 teams for $\beta \in \{10^3, 10^4, 10^5\}$. For example, a run on an instance with 12 teams and a beam width of 10^5 takes roughly 10 hours. We believe it is possible to improve this further by an order of magnitude using a compiled language.

8 Conclusion and Future Work

We investigated a beam search approach for the well-known traveling tournament problem. To this end, we proposed a recursive state space formulation, which is searched by a restricted breadth-first-search. This beam search is implemented in a memory efficient variant allowing for high beam widths to be tested. For guiding the search, we studied different lower bounds derivable from a state. Furthermore, we introduced a randomized beam search variant which applies parameterized Gaussian noise to the state ordering heuristic in order to diversify the search when performing multiple runs in parallel. We contribute

Table 5. Comparison of the final solution lengths of parallel randomized beam search using either lexicographic team ordering or random team ordering (RTO) with 30 independent runs each, parameters $\sigma_{\mathrm{rel}} = 0.001$, $\beta = 10^5$, and the CVRPH lower bound function (RBS-CVRPH) with the reported solution lengths of ant-colony optimization (AFC-TTP) [11], composite-neighborhood tabu search (CNTS) [3], simulated annealing (TTSA) [1], and population-based simulated annealing (PBSA) [14], where the latter is either used from scratch (PBSAFS) or starting from an already high quality solution (PBSAHQ) provided by a TTSA run. †New best feasible solutions.

inst	RBS-CVRPH		RBS-CVRPH-RTO		AFC-TTP		CNTS		TTSA		PBSAFS		PBSAHQ	
	min	mean	min	mean	min	mean	min	mean	min	mean	min	mean	min	mean
nl12	112680	113594.6	112791	113581.5	112521	114427.4	113729	114880.6	112800	113853.0	**110729**	**112064.0**	n/a	n/a
nl14	192625	198912.6	196507	199894.8	195627	197656.6	194807	197284.2	190368	192931.9	**188728**	190704.6	**188728**	**188728.0**
nl16	266736	271367.1	265800	270925.9	280211	283637.4	275296	279465.8	267194	275015.9	**261687**	265482.1	262343	**264516.4**
circ12	410	415.7	410	414.6	430	436.0	438	440.4	n/a	n/a	**404**	418.2	408	**414.8**
circ14	632	641.0	**630**†	**640.7**	674	692.8	686	694.4	n/a	n/a	640	654.8	632	645.2
circ16	918	933.8	**910**†	931.6	1034	1039.6	1016	1030.0	n/a	n/a	958	971.8	916	**917.8**
circ18	1300	1322.0	1296	1320.4	1486	1494.8	1426	1440.8	n/a	n/a	1350	1371.6	**1294**	**1307.0**

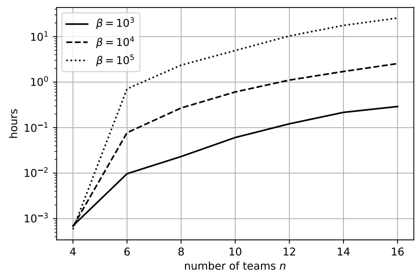


Fig. 3. Runtimes in hours for deterministic beam search runs on NL instances with $\beta \in \{10^3, 10^4, 10^5\}$.

.

a method based on decision diagrams to pre-calculate the existing capacitated vehicle routing problem and minimum number of trips bounds for instances up to 18 teams and how these bounds can be effectively used for any given state. To compare different lower bounds and tune algorithmic parameters, we created artificial instances. This allowed us to ultimately achieve better results on difficult NL and CIRC benchmark instances than the also mainly constructive ant-colony optimization approach AFC-TTP and the composite-neighborhood tabu search CNTS. For the circular instances with 14 and 16 teams we could find new best feasible solutions. Overall, the simulated annealing based approaches TTSA/PBSA still remain dominant.

We have implemented our approach as a prototype in Python 3.7. A re-implementation in a compiled language is desirable as much better runtimes and smaller memory footprints can be expected, which would allow to tackle even higher beam widths. So far we did not consider any local search, but a natural extension would be to try to further improve a number of best solutions provided by our beam search by local search. Furthermore, the provided state space formulation and lower bound methods might also be incorporated into GRASP or ACO algorithms, as well as into exact techniques such as A* variants.

To tackle instances with more than 18 teams with lower bound guidance, an interesting direction could be to use relaxed decision diagram for the bound precalulcations, in order to keep the memory and computational demand reasonably bounded.

References

1. Anagnostopoulos, A., Michel, L., Van Hentenryck, P., Vergados, Y.: A simulated annealing approach to the traveling tournament problem. Journal of Scheduling **9**(2), 177–193 (2006)
2. Bergman, D., Cire, A.A., van Hoeve, W.J., Hooker, J.N.: Decision Diagrams for Optimization. Artificial Intelligence: Foundations, Theory, and Algorithms, Springer (2016)
3. Di Gaspero, L., Schaerf, A.: A composite-neighborhood tabu search approach to the traveling tournament problem. Journal of Heuristics **13**(2), 189–207 (2007)
4. Easton, K., Nemhauser, G., Trick, M.: The traveling tournament problem description and benchmarks. In: International Conference on Principles and Practice of Constraint Programming. LNCS, vol. 2239, pp. 580–584. Springer (2001)
5. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3**, 43–58 (2016)
6. Ow, P.S., Morton, T.E.: Filtered beam search in scheduling. International Journal of Production Research **26**(1), 35–62 (1988)
7. Rasmussen, R.V., Trick, M.A.: A benders approach for the constrained minimum break problem. European Journal of Operational Research **177**(1), 198–213 (2007)
8. Ribeiro, C.C., Urrutia, S.: Heuristics for the mirrored traveling tournament problem. European Journal of Operational Research **179**(3), 775–787 (2007)
9. Thielen, C., Westphal, S.: Complexity of the traveling tournament problem. Theoretical Computer Science **412**(4-5), 345–351 (2011)
10. Urrutia, S., Ribeiro, C.C., Melo, R.A.: A new lower bound to the traveling tournament problem. In: 2007 IEEE Symposium on Computational Intelligence in Scheduling. pp. 15–18. IEEE (2007)
11. Uthus, D.C., Riddle, P.J., Guesgen, H.W.: An ant colony optimization approach to the traveling tournament problem. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation. pp. 81–88. ACM (2009)
12. Uthus, D.C., Riddle, P.J., Guesgen, H.W.: DFS* and the traveling tournament problem. In: International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems. LNCS, vol. 5547, pp. 279–293. Springer (2009)
13. Uthus, D.C., Riddle, P.J., Guesgen, H.W.: Solving the traveling tournament problem with iterative-deepening A*. Journal of Scheduling **15**(5), 601–614 (2012)
14. Van Hentenryck, P., Vergados, Y.: Population-based simulated annealing for traveling tournaments. In: Proceedings of the 22nd National Conference on Artificial Intelligence. pp. 267–262. No. 1, MIT Press (2007)