Too Constrained for Genetic Algorithms too Hard for Evolutionary Computing the Traveling Tournament Problem

Kristian Verduin[1][a], Sarah L. Thomson[2][b] and Daan van den Berg[1][c]
[1]*Department of Computer Science, Vrije Universiteit Amsterdam and Universiteit van Amsterdam, The Netherlands*
[2]*Edinburgh Napier University, U.K.*

Keywords: The Traveling Tournament Problem, Constraints, Genetic Algorithms, Evolutionary Computing, Constraint Hierarchy.

Abstract: Unlike other NP-hard problems, the constraints on the traveling tournament problem are so pressing that it's hardly possible to randomly generate a valid solution, for example, to use in a genetic algorithm's initial population. In this study, we randomly generate solutions, assess the numbers of constraint violations, and extrapolate the results to predict the required number of samples for obtaining a single valid solution for any reasonable instance size. As it turns out, these numbers are astronomical, and we finish the study by discussing the feasibility of efficient sampling of valid solutions to various NP-hard problems.

1 INTRODUCTION

A long long time ago, before computers existed, American kids played outside to entertain themselves. Often baseball, a true American invention, which over the years grew into the number #3 sport on the continent, having 19.1 million practitioners, and over 35 million TV spectators on a peak day (Bas, 2023). With such enormous numbers of engagement, North America is host to Major League Baseball (MLB), founded in 1876 and consisting of 30 teams from the United States of America and Canada, holding broadcasting rights involving billions of dollars (Solberg and Gaustad, 2022). It is therefore no surprise that scheduling the matchups of teams is considered a crucial part of the process.

What *is* a surprise though, is that until relatively recently, this was done *by hand* by Henry and Holly Stephenson, a married couple from Massachusetts. Perhaps most astonishing is the level of expertise these people must have possessed in creating these schedules, especially retrospectively considering the hardness of the problem. In the day (and maybe still), teams, players, coaches and unions filed a host of complicating constraints: "Red Sox must be home on Patriot's Day", "Blue Jays must be home on Canada Day", "a team on the road on Memorial Day must be home on the 4^{th} of July" are just some of the demands the Stephensons processed – by mail – and incorporated into the huge scheduling task the MLB presented. But as a typical 21^{st}-century's informatics history, computers came along and in 2004, after 24 years of manual labour, the Stephensons were outbid by the Sports Scheduling Group, who teamed up with Carnegie Melon and Georgia Tech (Press, 2004).

In time, most if not all human labour will likely be outclassed by machines, but the timing of the appearance of MLB's first planning algorithm might not have have been completely coincidental. Just a few years earlier, Easton, Nemhauser and Trick formulated the *Traveling Tournament Problem* (TTP)[1], which entails scheduling tournament rounds of an even number of teams (n_{teams}) (Easton et al., 2001). In TTP, each team must play each other team *twice* in the schedule (once at home, once away), which is known as the **double round-robin** constraint. Additionally, when team A plays team B in one round, the exact inverse match (B playing A) cannot take place in the consecutive round, which is known as the **noRepeat** constraint. Finally, there is the maximum number of consecutive games any team can play at

[a] https://orcid.org/0009-0005-8754-7635
[b] https://orcid.org/0000-0001-6971-7817
[c] https://orcid.org/0000-0001-5060-3342

[1]not the to be confused with the Traveling Thief Problem, also abbreviated to TTP, which is clearly the fault of Markus Wagner.

Verduin, K., Thomson, S. and van den Berg, D.
Too Constrained for Genetic Algorithms too Hard for Evolutionary Computing the Traveling Tournament Problem.
DOI: 10.5220/0012192100003595
In *Proceedings of the 15th International Joint Conference on Computational Intelligence (IJCCI 2023)*, pages 246-257
ISBN: 978-989-758-674-3; ISSN: 2184-3236

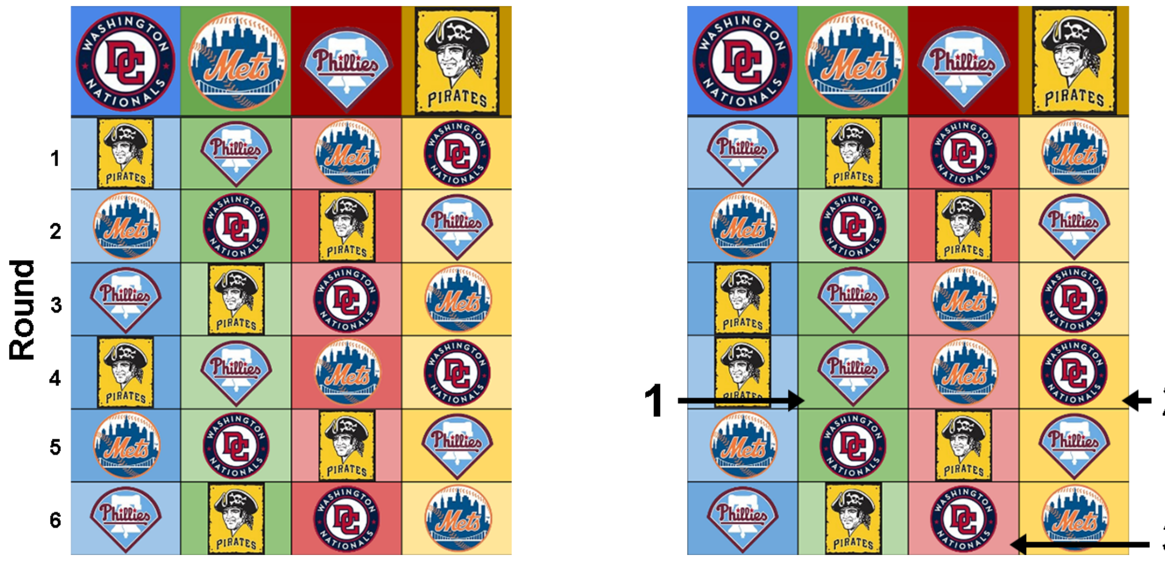

Figure 1: **Left:** A valid TTP schedule, with lighter backgrounds designating home games, and darker away. **Right:** An invalid TTP schedule that has all three constraints violated: *doubleRoundRobin* (1) as the Mets play the Phillies at home *twice* in the schedule, *noRepeat* (2) and *maxStreak* (3). Note that point (1) has a *noRepeat* violation, in addition to the *doubleRoundRobin* violation.

home or away, the **maxStreak** constraint[2]. Usually, *maxStreak* = 3 meaning any team can at most play three consecutive rounds at home, or away, anywhere in the schedule (Thielen and Westphal, 2011). Only three constraints, but they can be violated many times per schedule, especially for larger numbers of n_{teams} (Figure 1). And as constraints go, it only takes one violation to render the entire schedule invalid.

But the actual problem is not about satisfying these constraints. The traveling tournament problem, like the traveling salesman problem (TSP), has a distance matrix which holds the travel time between stadiums (or: cities), and the main task is to minimize the total travel time. As we will shortly see however, these three constraints are so asphyxiating, that travel time optimization almost becomes auxiliary to finding a valid schedule in the first place.

From here, we will first inspect the complexity timeline of the TTP in Section 2. It took some time before the 2001-formulation of the problem was fully proved to be NP-complete, quite understandably as these proofs are usually not very easy to construct. In Section 3, we will discuss some of the algorithmic approaches to TTP. This history is quite uncommon, as it has relatively few metaheuristic applications, but quite a number of approximation algorithms for various variants as well as for the seminal formulation. After that, the experiment and the results are laid out in Sections 4 and 5. Finally, in Section 6 we discuss how to interpret these results: can NP-hard optimization problems be ranked on their constraints? We make some suggestions for a classification.

[2]Terminology varies slightly across literature.

2 COMPLEXITY TIMELINE OF THE TTP

The traveling tournament problem is hard. Its formal classification history starts in 2009 with a non-peer reviewed preprint by Rishiraj Bhattacharyya who showed that every instance of the traveling *salesman* problem can be polynomially converted into an instance of the traveling *tournament* problem (Bhattacharyya, 2009). Thus, if the traveling tournament could have been solved exactly in polynomial time, so could the traveling salesman problem. The latter is not the case: the traveling salesman problem requires *exponential* time to solve exactly, and therefore, so does the traveling tournament problem[3]. This initial proof had a loose end though: Bhattacharyya's proof involved the TTP without the *maxStreak* constraint, and even though he states "even without" this constraint the problem is NP-hard, there was no guarantee that addition of this property would not actually make the problem easier.

But as often happens in science, intuition precedes proof, and it turned out Bhattacharyya was right. History threw a curveball though, as the proof

[3]all assuming $P \neq NP$, which appears to be the majority consensus in the scientific community.

of full NP-completeness (including the $maxStreak = 3$ constraint as originally formulated by Easton & Trick) appeared in 2011, not by a reduction from TSP, but by a 'classic' reduction from satisfiability (SAT) (Thielen and Westphal, 2011). Bhattacharyya's original 'streakless' from-TSP-proof was nonetheless peer reviewedly published in 2015 by Operations Research Letters (Bhattacharyya, 2016). As a side note, the terms NP-hard and NP-complete were apparently used interchangeably in these studies.

Another 5 years later, in 2021, Diptendu Chatterjee further widened the insight by proving that indeed for every $maxStreak > 3$, the problem is NP-hard. Although the manuscript will undoubtedly walk a similar path to a highly regarded journal, it is currently only available as an arXiv preprint (Chatterjee, 2021). The takeaway observation here is that it took at least 12 and at most 22 years of hard intellectual work to obtain a full proof of NP-hardness for the TTP. The historic trajectory itself is also interesting, as the first inning encompassed only a 'no-maxStreak proof', the second inning a slightly wider but totally different proof for $maxStreak = 3$, but still a third inning was needed for the full proof with $maxStreak > 3$. Furthermore, the constructed proofs differed substantially, reducing from TSP, SAT and k-SAT respectively, and we hypothesize that for these reasons, it is quite likely that the *maxStreak*-constraint plays a key role not only in the problem's hardness classification, but also in the arduous task of solving its individual instances.

In any case, one NP-hard problem is not the other. Being in the same class might suggest hardness equivalence, which might be true in a general sense, but is certainly questionable when looked at with any degree of resolution; the difficulty of finding an optimal solution to an integer partition problem instance (van den Berg and Adriaans, 2021), a traveling salesman problem instance (Liang et al., 2022; Sleegers et al., 2020) or a traveling tournament problem instance is very different in practice. One complicating aspect seems to arise from the constraints of a problem and for TTP, these are pretty severe. In fact, we will argue, it is almost impossible to generate a population of valid random individuals by the most basic procedures, simply because its constraints invalidate the vast majority of instances.

3 ALGORITHMS FOR THE TTP

Could it be for these reasons, the asphyxiating constraints on the traveling tournament problem itself, that genetic algorithms are almost completely absent from its algorithmic history? Possibly. A study from 2023 showed that even for a very small instance size of $n_{teams} = 30$, which corresponds closely to the MLB's actual number of teams, the number of constraint violations in a randomly generated schedule is close to 3800, and increases quadratically in n_{teams} (Verduin et al., 2023). This means it is nontrivial to create an initial population of valid individuals for a genetic algorithm to work with. We would like to point out to the reader that the situation is quite different for the "closely related" traveling salesman problem (in the same complexity class, and one reduces the other) which has seen plethora of population-based algorithms applied to it (Eiben et al., 2003; Koppenhol et al., 2022; Potvin, 1996).

Still, a few hill climbing and simulated annealing approaches have been applied to the TTP; these could both be regarded as evolutionary algorithms with a population size of 1 (Anagnostopoulos et al., 2006; Lim et al., 2006; Jha and Menon, 2014) (the last one applied to the *mirrored* traveling tournament problem, a special, possibly easier variant of TTP). Also, there is a non-evolutionary approach with ant colony optimization (Uthus et al., 2009), tabu search (Gaspero and Schaerf, 2007), an iterative 'three strike algorithm' (Ruth et al., 2023), and some integer programming in an experiment done by the problem's founders themselves (Easton et al., 2003). A rather complex bound-driven version of beam search was developed by Nikolaus Frohner and his colleagues, augmented with a hint of randomness (Frohner et al., 2020). Nonetheless, optimal solutions found in these experimental setups usually range from $n_{teams} = 10$ to $n_{teams} = 16$, a staggeringly small number, especially when compared to the Euclidean traveling salesman problem, which can be solved exactly by a (non-general) approach up to many thousands of cities (Applegate et al., 2009).

There are a few internet documents reporting application of a genetic algorithm to the TTP. Most of these are either non-scientific papers, or provide so little explanation on their methods that one could better start anew. One notable exception is the work by Meriem Khelifa and her team(Khelifa et al., 2017). As a rare exception, these authors do report a full-fledged genetic algorithm for the TTP. It is rather complicated, and utilizes mutations similar to the simulated annealing approach by Lim et al. (Lim et al., 2006). Both approaches use construction of initial valid schedules which might not be uniformly random though, and whether the neighbourhood definitions facilitate homogeneous exploration of the combinatorial space, such as is the case with n-opt in traveling salesman, also remains to be seen.

Quite interestingly, despite the scarcity of metaheuristic approaches to the TTP, its algorithmic history does in fact sport quite a number of approximation algorithms, which are in some sense of nobler blood. Approximation algorithms are usually more difficult to program, run in polynomial time, but different from metaheuristics also give a *guarantee* on their solution quality, usually given as a ratio (or 'deficiency') on the optimum. As an example, the Christofides-Serdyukov algorithm for the traveling salesman problem is a 1.5-approximation algorithm, which means that it returns a solution for a TSP-instance in polynomial time that is *at most* 50% longer than the optimal tour (Christofides, 1976; Serdyukov, 1978; Karlin et al., 2021). Pretty good when considering that an exact solution (which would be equivalent to a deficiency of 1) requires exponential time – since the problem is NP-hard. But although the TTP is arguably harder than the TSP, it does have a number of approximation algorithms. Quite a few actually.

On first base, Clemens Thielen and Stephan Westphal published an $(3/2+6/(n-4))$- approximation algorithm for TTP with $maxStreak = 2$ in 2010 (Thielen and Westphal, 2010), and in 2012, the same authors expanded further on the problem in a publication that might be an extension to the former (Thielen and Westphal, 2012). For a more general case, Stephan Westphal teamed up with Karl Noparlik to produce a 5.875-approximation algorithm, which was published in 2014 (Westphal and Noparlik, 2014). As a forte, the approximation ratio is constant, but only for $n_{teams} \geq 6$ and $maxStreak \geq 4$.

On second base, we have the approximation algorithm by Miyashiro et al. from 2010 (Miyashiro et al., 2012). It is possibly the first approximation algorithm on TTP, and has a ratio of at most $2+(9/4)/(n_{teams}-1)$, but the same team expanded with Daisuke Yamaguchi published an improved version exactly one year later (Yamaguchi et al., 2011). Interestingly, the performance of the improved algorithm depends on the *maxStreak*-parameter, which we intuitively suspect to play a central role in the TTPs constraint-hardness. Three years later, in 2014, the same team published a 2.75-approximation algorithm for the TTP (Imahori et al., 2014). A flummoxing low ratio, but the algorithm in this case applied to the 'unconstrained TTP', which had its *maxStreak* and *noRepeat* constraints removed.

On third base are teams incorporating MingYu Xiao. In 2016, he and XiaoWei Kou published an approximation algorithm for the incredibly constraining variant of TTP with $maxStreak = 2$, providing a ratio of $(1+4/n)$, which is very low (Xiao and Kou, 2016). He improved the ratio to $1+O(1/n)$ with coauthor JingYang Zhao in 2021, for TTP with $n_{teams}/2$ being odd (Zhao and Xiao, 2021). The most recent addition to their winning streak is a 2022 paper, this time for $maxStreak = 3$, improving the approximation ratio to $1.598+\varepsilon$, which is remarkably close to Christofides-Serdyukov ratio of 1.5 for Euclidean TSP (Zhao et al., 2022).

So despite the scarcity of evolutionary algorithms for the TTP, there is a widely developing outfield of approximation algorithms, the best of which provide very promising ratios. Whoever hopes for the home run of a 1-approximation algorithm, however, should realize that this is very unlikely to happen. A 1-approximation for the TTP would mean an optimal solution in polynomial time, which can only happen if TTP turns out to be in P instead of in NP. From mutual reducibility, this would mean that the entire class of NP would cease to exist, contradicting the community's consensus that probably $P \neq NP$. Having said this, we do hope that the aforementioned teams keep lowballing the approximation ratios, and are curious to see whether the magical Christofides bound of 1.5 will be reached, or even broken for TTP as well.

4 EXPERIMENT

To gain insight into the numbers and distributions of violatable constraints, we devised a naive but fast procedure that generated 5×1 million random schedules for each $n_{teams} \in \{4, 6, 8...46, 48, 50\}$. It did not take into account any of the three constraints while generating the schedule, but rather recorded the constraint violations retrospectively. The reason for making $5 \times$ 1 million random schedules lies in the practicality of getting a somewhat higher resolution of the initial curvature in Figure 2, as well as and a reasonable aspect ratio; theoretically, it should not make a difference.

A complete schedule for the TTP consists of $2(n-1)$ rounds, each of which is randomly filled up by randomly selecting an opponent, after which one is designated as 'home' and the other 'away'. The opponent is then marked as 'placed' and cannot be selected again in the same round. Round by round, the entire schedule is filled up this way, which can be done in linear time. Note that although the method completely disregards the three TTP constraints (*doubleRoundRobin*, *maxStreak*, *noRepeat*) it is still tighter than a *complete* random fill: by using the 'augmented permutation', we ensure that every team has exactly one opponent per round, pretty much like a permutation in a TSP-instance ensures every sequence is a full tour. Also, teams cannot play

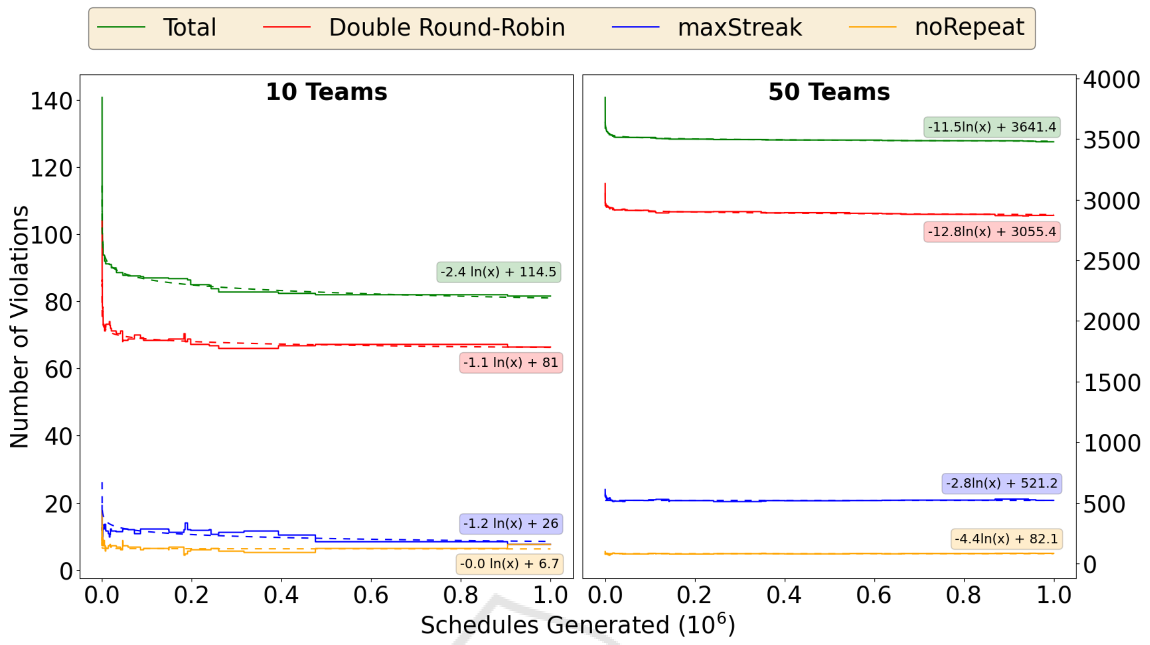

Figure 2: A run of one million randomly generated TTP schedules, retaining the best sample at each generation, produced progressively better schedules, but failed to generate a single valid schedule for any value $n_{teams} > 4$. Shown are values for 10 teams (left subfigure) or 50 teams (right subfigure), averaged over 5 runs, and characterized by a logarithmic function fit (dashed lines).

themselves, and each match holds one home and one away team.

After filling up the schedule, the constraint violations are counted. If team A plays team B more than twice, each game after the second counts as an additional *doubleRoundRobin* violation. If team A plays team B only once, it is also counted as an additional *doubleRoundRobin* violation. If team A does not play team B exactly once at home and once away, an additional *doubleRoundRobin* violation is added. If team A plays at their home or away venue more than three games consecutively, each game after the third counts as an additional *maxStreak* violation on the schedule. Finally, if teams A and B play each other in more than one consecutive round, each consecutive game after the first counts as an additional *noRepeat* violation.

For each $n_{teams} \in \{4, 6, 8 \dots 46, 48, 50\}$, five runs of one million randomly generated schedules were completed, summing up to 120 million schedules. Note that is is technically not impossible to look at uneven values for n_{teams}, but a ‘rest’ would be inserted as an empty entry, which would be assigned twice for every team in a double round-robin schedule. From the 120 million schedules, we took the following data:

1. For a single run with n_{teams}, we kept track of the lowest number of violations for generated schedules in a run. The idea was to find out how many randomly generated schedules are needed for one valid (zero-violations) schedule. Such random valid schedules might then be used in creating an initial population for evolutionary algorithms. This hope however, turned out to be idle.
Nonetheless this data facilitated a forward projection. From the five runs, the average was taken on each of the million points, through which a monotonically decreasing function $a \cdot ln(x) + b$ was fit. From these characterizations, the required number of random samples to obtain just one valid schedule was projected for all numbers of n_{teams}, by calculating $a \cdot ln(x) + b = 0$ (see columns 3, 4 and 5 in Table 1).

2. For each n_{teams}, all 5 million samples were collectively histogrammed, showing distributions for all 3 violation types, as well as the total number of violations. These histograms were then characterized by the fit of a normal distribution, which was then used to assess the probability of sampling a zero-violation schedule. As the normal distribution is continuous, we took the chance of the number of conflicts to be under 0.5 (see second column in Table 1). Although these values should correspond to the results of the previous method, we did not try to reconcile them, as a tiny difference in fits might result in a huge difference in extrapolation values. Rather, we think

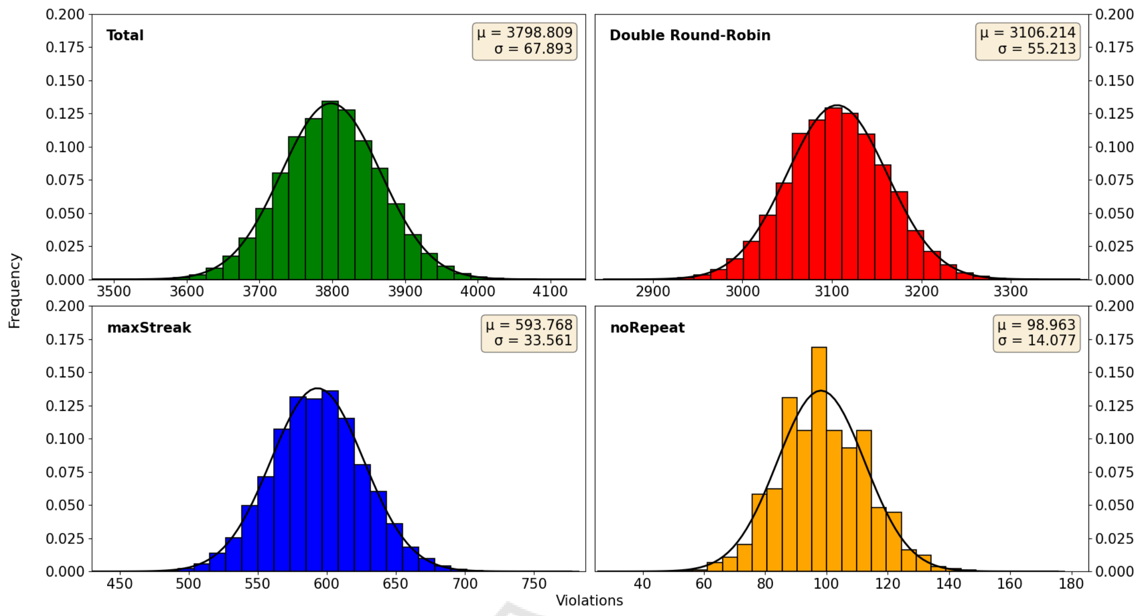

Figure 3: Distribution of constraint violations for the traveling tournament problem. An average randomly constructed schedule for 50 teams has no less than 3799 violations, of which 3106 round-robin violations, 594 *maxStreak*-violations and 99 *noRepeat* violations (clockwise from top-left respectively).

both measurements will tell the same story from a different angle: making random schedules hoping for a valid result is idle. Still, calculating one from the other might give inconsistent results from reasons of precision. Fits were made with the *scipy.optimize.curve_fit*-package in python.

3. For all n_{teams}, the minimum, maximum and average for all 3 types of constraint violations was characterized by fitting a quadratic function function through n_{teams}, showing the expected number for each type of constraint violation, and how these expected numbers grow through n_{teams}. Results can be seen in Figure 4.

All generation was done in 24 threads on SURF's Lisa Compute Cluster[4] running Debian Linux in approximately one day, and our Python source code is publicly available (Anonymous, 2023).

5 RESULTS

5.1 Numbers of Required Samples

A million samples in a run, retaining the best schedule (having the fewest violations) turns out to be hugely insufficient for generating anything even remotely valid. Only for 4 teams, zero-violation schedules were found: 606, 618, 619, 649 and 656 times respectively for each of its 5 runs – an average of just 0.06%. For 10 teams, the number of violations never dropped below 79, and for 50 teams, no value lower than 3458 violations was ever found throughout the 5 runs of 1 million samples (Figure 2). The R^2 values for the total number of violations *doubleRoundRobin* were quite bad, over 0.63, possibly due to the high initial segment. The R^2 on *noRepeat* was below 0.003 for both *nTeams*, possibly for the inverse reason.

There appears to be some degree of independence between the violation types. As can be seen in the right subfigure of Figure 2, the numbers of *maxStreak*-, *noRepeat*- and *doubleRoundRobin*-violations can all temporarily increase even when the total number of violations goes down. A future study could address this issue; if it turns out that one type of violation can be minimized with little or no effect on the others, it might provide a method to generate (random) valid schedules much faster. Intuitively, maybe the *noRepeat* constraint can be resolved last.

5.2 Distributions of Violations

For each number of n_{teams}, 5 million randomly sampled schedules produced near-normal distributions for *maxStreak*-violations, *noRepeat*-violations, *doubleRoundRobin*-violations, and the total number of constraint violations (Figure 3 shows the values for

[4]https://www.surf.nl/en/lisa-compute-cluster-extra-processing-power-for-research

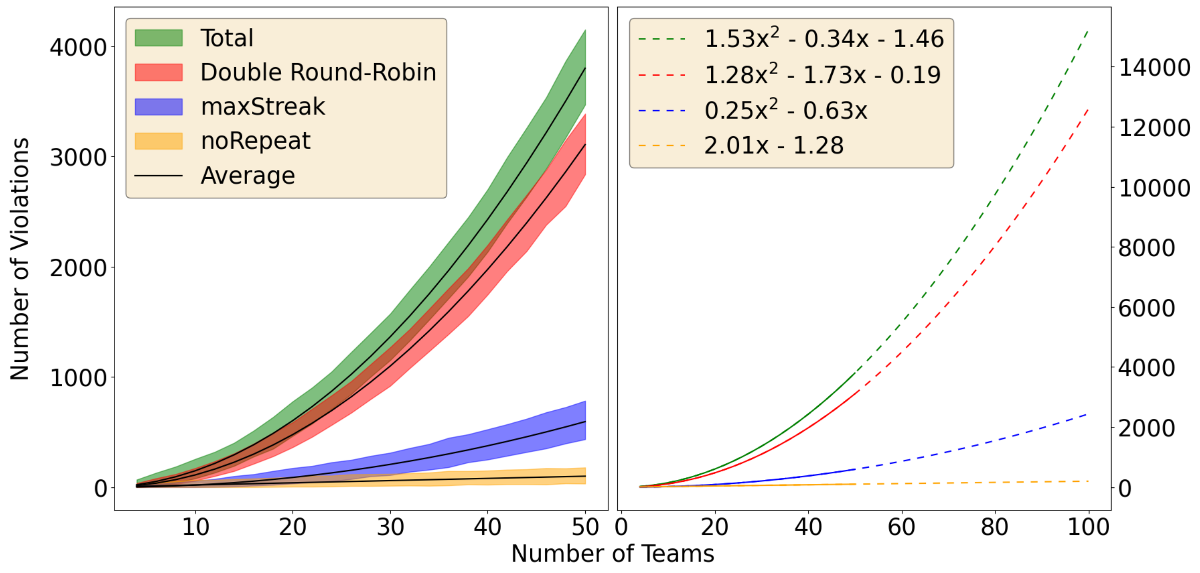

Figure 4: **Left:** In creating random solutions for the traveling tournament problem, expectancy for most constraint violations increases quadratically through n_{teams}; only the *noRepeat* violations increase linearly. The color-filled area is the spread of the violations (between max and min). **Right:** Dashed lines are the extrapolations of the violations.

50 teams).

The first thing that stands out is the fickleness of the *noRepeat*-constraint. In figure 3, the fit on the *noRepeat* constraint has an RMSE of 0.012 while other fits have an error of at most 0.002 – quite a difference. Though it would be easy to attribute the bad fit to the relatively low occurrence of the violation, something more peculiar seems to be going on. When assessing the error of the fits of the normal distributions, they consistently decrease as n_{teams} increases. This is expected, as a larger range of values accommodates for smoother distributions, and therefore lower fit errors. But while this is true for the fit errors on *doubleRoundRobin*-, *maxStreak*-, and total number of violations, which drop below 0.01 for all values of $n_{teams} \geq 14$, fits on the *noRepeat*-violations only drop below 0.01 three times in the range $4 \leq n_{teams} \leq 50$, at values $n_{teams} \in \{34, 38, 40\}$. A provisional, somewhat unsatisfying explanation might be found in the fact that the *noRepeat* violations 'only' increase linearly through n_{teams} while violations on *doubleRoundRobin* and *maxStreak* (and thereby also the total number of violations) increase quadratically (Figure 4). In any case, the number of *noRepeat* violations appears to stabilize much slower into a normal distribution, and behaves more erratic for all n_{teams}, whereas all other violation types more clearly stabilize, with progressively lower fit errors.

5.3 Violations Grow Quadratically

From the violation distributions, the maximum, minimum and average values for each type of constraint violation were recorded for each n_{teams}, after which a quadratic function was fit to the average. On the left side of Figure 4, the filled area shows the spread of violations whereas the solid line gives the average. On the right side, the fitted quadratic functions are shown, with extrapolated values in dashed lines. Both the *doubleRoundRobin* violations and *maxStreak* violations show quadratic growth with n_{teams}, as $1.22x^2 - 1.69x - 0.18$ and $0.25x^2 - 0.63x$ respectively. The *noRepeat* violations increase linearly with n_{teams} as $2.01x - 12.27$. All fits are tight, with $RMSE \leq 0.06$ for all fitted functions. Results similar to these were reported earlier (Verduin et al., 2023), with slightly different function parameters.

6 CONCLUSION & DISCUSSION

It appears that the semi-naive (but really fast) way of creating random initial schedules is unusable for valid schedules of any reasonable n_{teams}. In this experiment, the minimum number of violations over 1 million random schedules with $n_{teams} = 10$ only was 79 – nowhere *near* valid. For $n_{teams} = 20$ this number had increased to 464, after which it exploded to 1151, 2134 and 3468 for $n_{teams} = 30$, 40 and 50 respectively. Trying to generate valid schedules by uniform ran-

Table 1: Time needed to generate a random valid schedule. The second column is the probability of generating a valid schedule from the fits of the normal distributions. The third column is the estimated number of samples needed for obtaining a valid schedule from the logarithmic fits. The fourth and fifth columns give a rough indication of required computation time.

n_{teams}	$p_{feasible}$	Req. samples	Req. time (sec)	Required time
4	2.10×10^{-3}	9.293×10^{-6}	1.369×10^{-9}	$<$ 2 nanoseconds
6	2.96×10^{-7}	2.439×10^{8}	7.693×10^{4}	$\approx$ 21 hours
8	1.21×10^{-13}	1.122×10^{13}	6.348×10^{9}	$>$ 200 years
10	2.51×10^{-22}	6.110×10^{20}	5.309×10^{17}	$>$ 16.8 billion years
12	1.18×10^{-31}	6.503×10^{27}	8.169×10^{24}	$> 10^{7} \times$ age universe
14	3.74×10^{-45}	1.074×10^{40}	1.852×10^{37}	
16	8.69×10^{-61}	1.253×10^{31}	2.834×10^{28}	
18	7.84×10^{-77}	8.463×10^{34}	2.468×10^{32}	
20	1.09×10^{-96}	3.995×10^{56}	1.404×10^{54}	
22	1.30×10^{-121}	6.842×10^{40}	2.883×10^{38}	
24	9.76×10^{-145}	3.142×10^{61}	1.566×10^{59}	
26	1.28×10^{-169}	3.572×10^{47}	2.046×10^{45}	
28	7.08×10^{-199}	1.375×10^{81}	9.129×10^{78}	
30	3.17×10^{-230}	2.100×10^{50}	1.579×10^{48}	$> 10^{30} \times$ age universe
32	1.94×10^{-266}	6.599×10^{61}	5.644×10^{59}	
34	2.28×10^{-301}	5.297×10^{68}	5.079×10^{66}	
36	≈ 0	3.225×10^{100}	3.632×10^{98}	
38	≈ 0	1.057×10^{76}	1.302×10^{74}	
40	≈ 0	1.104×10^{63}	1.544×10^{61}	$> 10^{43} \times$ age universe
42	≈ 0	3.181×10^{95}	4.858×10^{93}	
44	≈ 0	5.583×10^{84}	9.359×10^{82}	
46	≈ 0	1.277×10^{95}	2.336×10^{93}	
48	≈ 0	9.694×10^{104}	1.946×10^{103}	
50	≈ 0	7.684×10^{138}	1.704×10^{137}	$> 10^{119} \times$ age universe

dom sampling knocks the computational budget completely out of the ballpark, with an estimated 16.8 billion years of computation time for $n_{teams} = 10$ on a single core, up to many times the age of the universe for $n_{teams} = 30$ (the MLB size), and rising ever faster beyond that, just to find a single valid schedule.

No wonder therefore that Anagnostopoulos et al. generate the initial schedule for their simulated annealing by backtracking by which "[valid] schedules were easily obtained" (Anagnostopoulos et al., 2006). That study only uses $n_{teams} \leq 16$, and as backtracking algorithms are of exponential time complexity, much larger numbers are probably undoable by their approach. Besides, these authors treat the *noRepeat* and *maxStreak* as 'soft constraints' and their mutations might turn valid schedules into invalid ones. So even for an algorithm as simple as simulated annealing, the constraints on the TTP appear problematic. This is also seen in Lim et al. (Lim et al., 2006), and the GA approach by Khelifa et al. (Khelifa et al., 2017).

Gaspero & Schaerf (Gaspero and Schaerf, 2007) and Ribeiro & Urrutia (Ribeiro and Urrutia, 2007) both represent the schedules using one-factorization of a graph. They then continue to generate a single round-robin schedule, which is mirrored and reversed to combine into a double round-robin. It should be noted that the subclass of mirrored TTP might be easier than regular TTP because its defined construction eliminates the need for the *noRepeat* constraint. It might also yield more expensive schedules than regular TTP, but nonetheless it is an often deployed technique in North American scheduling practice (Ribeiro and Urrutia, 2007). Lim et al. (Lim et al., 2006) (also) generate their initial solutions with the use of a three-phase approach. These authors also mirror their solution, and argue that doing so greatly reduces the required time for finding a valid solution. Maybe the lesson here is that sacrificing some quality in exchange for validity is not a bad tradeoff.

Combining these findings with the results presented in our study, the following conjectures spring forth:

- The traveling tournament problem has severe constraints, making it hard to either to randomly generate solutions uniformly, or to mutate one solution to a TTP instance into another, as many mutation types might turn valid schedules into invalid ones. For the available mutation types from literature, the explorability of the resulting neighbourhood structure should be explored.
- It is harder to find randomized valid solutions for the traveling tournament problem than for other NP-hard problems. This is unlike the number partition problem, for which any random allocation of an integer to either subset is valid and done in linear time (van den Berg and Adriaans, 2021; Sazhinov et al., 2023). This is unlike TSP, for which any permutation of cities yield a new subset of edges that forms a closed loop, and thereby a valid solution. This is unlike the Job Shop Scheduling Problem, for which most permutations, after a construction phase, yield a new and valid solution (de Bruin et al., 2023; Weise et al., 2021) . This is a bit like 2D protein folding though, in which one can constructively generate a random solution with relatively little backtracking, but mutating from one solution to the next is still problematic (van Eck and van den Berg, 2023; Jansen et al., 2023; Jamil and Yang, 2013). TTP is on the higher end of this list, with only (stochastic) backtracking to guarantee a valid uniform random initial solution, and mutation being highly problematic.
- Although all problems mentioned in the previous point are NP-hard, maybe they can be classified into a hierarchy of constructibility, in which the lower end of the scale holds problems with instances for which randomized solutions are readily created, and easily mutated into other solutions, while the higher end of the scale holds problems for which the generation of valid solutions takes a lot of time (either stochastically or deterministically). As a complete swing for the fences, we suggest a hierarchy on the complexity of the fastest algorithm that generates a random initial solution from all valid solutions with equal probability. A random valid solution to the partition problem or the TSP can be made in linear time, whereas a random valid solution to protein folding or the TTP might require exponential time in the worst case. We are unaware whether such a hierarchical classification already exists, and which place would be taken by the TTP.

7 STATEMENT

A pilot study on the quadratic fits for the number of conflicts has earlier been published as a late-breaking abstract at EvoSTAR (Verduin et al., 2023), for which the copyright lies with the authors. Although all experiments have been redone, scaled up, this part of the study might look somewhat similar. All other experiments are new, as are the figures and the background study on the problem.

ACKNOWLEDGEMENTS

We would like to express our respect to Henry and Holly Stephensons, the married couple that have scheduled the MLB baseball season for many years by hand. I hope that one day, I may talk to you and hear some of your stories.

A wink to our great colleague Markus Wagner, author of many papers on the traveling thief problem (Chagas and Wagner, 2020; Sachdeva et al., 2020; Yafrani et al., 2022; Chagas et al., 2021; Chagas and Wagner, 2022b; Yafrani et al., 2017; Yafrani et al., 2018; Wu et al., 2018; Martins et al., 2017; El Yafrani et al., 2018; Wu et al., 2017; Wagner, 2016; Faulkner et al., 2015; Wagner et al., 2018; Polyakovskiy et al., 2014). The traveling thief problem should be abbreviated to "TThP"; that way it would also be more aligned with the Thief Orienteering Problem, which is abbreviated to "ThOP" (Chagas and Wagner, 2022a). "Thief" is should be "Th".

REFERENCES

(2023). Sports in the united states. *Wikipedia*. Consulted June 21st, 2023.

Anagnostopoulos, A., Michel, L., Hentenryck, P. V., and Vergados, Y. (2006). A simulated annealing approach to the traveling tournament problem. *Journal of Scheduling*, 9:177–193.

Anonymous (2023). Repository containing source material: https://anonymous.4open.science/r/TTPSchedules-EE40/README.md.

Applegate, D. L., Bixby, R. E., Chvátal, V., Cook, W., Espinoza, D. G., Goycoolea, M., and Helsgaun, K. (2009). Certification of an optimal tsp tour through 85,900 cities. *Operations Research Letters*, 37(1):11–15.

Bhattacharyya, R. (2009). A note on complexity of traveling tournament problem. *Optimization Online*, 2480.

Bhattacharyya, R. (2016). Complexity of the unconstrained traveling tournament problem. *Operations Research Letters*, 44(5):649–654.

Chagas, J. B., Blank, J., Wagner, M., Souza, M. J., and Deb, K. (2021). A non-dominated sorting based customized random-key genetic algorithm for the bi-objective traveling thief problem. *Journal of Heuristics*, 27(3):267–301.

Chagas, J. B. and Wagner, M. (2020). Ants can orienteer a thief in their robbery. *Operations Research Letters*, 48(6):708–714.

Chagas, J. B. and Wagner, M. (2022a). Efficiently solving the thief orienteering problem with a max–min ant colony optimization approach. *Optimization Letters*, 16(8):2313–2331.

Chagas, J. B. and Wagner, M. (2022b). A weighted-sum method for solving the bi-objective traveling thief problem. *Computers & Operations Research*, 138:105560.

Chatterjee, D. (2021). Complexity of traveling tournament problem with trip length more than three. *arXiv preprint arXiv:2110.02300*.

Christofides, N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group.

de Bruin, E., Thomson, S. L., and Berg, D. v. d. (2023). Frequency fitness assignment on jssp: A critical review. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 351–363. Springer.

Easton, K., Nemhauser, G., and Trick, M. (2001). The traveling tournament problem description and benchmarks. In *Principles and Practice of Constraint Programming—CP 2001: 7th International Conference, CP 2001 Paphos, Cyprus, November 26–December 1, 2001 Proceedings 7*, pages 580–584. Springer.

Easton, K., Nemhauser, G., and Trick, M. (2003). Solving the travelling tournament problem: A combined integer programming and constraint programming approach. *Lecture notes in computer science*, pages 100–112.

Eiben, A. E., Smith, J. E., et al. (2003). *Introduction to evolutionary computing*, volume 53. Springer.

El Yafrani, M., Martins, M., Wagner, M., Ahiod, B., Delgado, M., and Lüders, R. (2018). A hyperheuristic approach based on low-level heuristics for the travelling thief problem. *Genetic Programming and Evolvable Machines*, 19:121–150.

Faulkner, H., Polyakovskiy, S., Schultz, T., and Wagner, M. (2015). Approximate approaches to the traveling thief problem. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 385–392.

Frohner, N., Neumann, B., and Raidl, G. (2020). A beam search approach to the traveling tournament problem. In *Evolutionary Computation in Combinatorial Optimization*, pages 67–82. Springer International Publishing.

Gaspero, L. D. and Schaerf, A. (2007). A composite-neighborhood tabu search approach to the traveling tournament problem. *Journal of Heuristics*, 13:189–207.

Imahori, S., Matsui, T., and Miyashiro, R. (2014). A 2.75-approximation algorithm for the unconstrained traveling tournament problem. *Annals of Operations Research*, 218(1):237–247.

Jamil, M. and Yang, X.-S. (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194. PMID: 55204.

Jansen, R., Horn, R., van Eck, O., Verduin, K., Thomson, S. L., and van den Berg, D. (2023). Can hp-protein folding be solved with genetic algorithms? maybe not. (submitted).

Jha, S. and Menon, V. (2014). Bbmttp: Beat-based parallel simulated annealing algorithm on gpgpus for the mirrored traveling tournament problem. In *Proceedings of the High Performance Computing Symposium*, pages 1–7.

Karlin, A. R., Klein, N., and Gharan, S. O. (2021). A (slightly) improved approximation algorithm for metric tsp. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 32–45.

Khelifa, M., Boughaci, D., and Aïmeur, E. (2017). An enhanced genetic algorithm with a new crossover operator for the traveling tournament problem. In *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 1072–1077. IEEE.

Koppenhol, L., Brouwer, N., Dijkzeul, D., Pijning, I., Sleegers, J., and Van Den Berg, D. (2022). Exactly characterizable parameter settings in a crossoverless evolutionary algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1640–1649.

Liang, T., Wu, Z., Lässig, J., van den Berg, D., and Weise, T. (2022). Solving the traveling salesperson problem using frequency fitness assignment. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 360–367. IEEE.

Lim, A., Rodrigues, B., and Zhang, X. (2006). A simulated annealing and hill-climbing algorithm for the traveling tournament problem. *European Journal of Operational Research*, 174(3):1459–1478.

Martins, M. S., El Yafrani, M., Delgado, M. R., Wagner, M., Ahiod, B., and Lüders, R. (2017). Hseda: a heuristic selection approach based on estimation of distribution algorithm for the travelling thief problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 361–368.

Miyashiro, R., Matsui, T., and Imahori, S. (2012). An approximation algorithm for the traveling tournament problem. *Annals of Operations Research*, 194:317–324.

Polyakovskiy, S., Bonyadi, M. R., Wagner, M., Michalewicz, Z., and Neumann, F. (2014). A comprehensive benchmark set and heuristics for the traveling thief problem. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 477–484.

Potvin, J.-Y. (1996). Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, 63:337–370.

Press, A. (2004). Husband-wife team out bid after 24 years. *ESPN online Numerical Optimisation*. (Consulted June 21st, 2023).

Ribeiro, C. C. and Urrutia, S. (2007). Heuristics for the mirrored traveling tournament problem. *European Journal of Operational Research*, 179(3):775–787.

Ruth, B., Jackson, S. J., and Mays, W. T. S. H. K. (2023). The threestrike algorithm for winning matches: Say it Ain't So Joe. *Journal of Baseball Algortihms*, 37(1):1101–1503.

Sachdeva, R., Neumann, F., and Wagner, M. (2020). The dynamic travelling thief problem: Benchmarks and performance of evolutionary algorithms. In *Neural Information Processing: 27th International Conference, ICONIP 2020, Bangkok, Thailand, November 18–22, 2020, Proceedings, Part V 27*, pages 220–228. Springer.

Sazhinov, N., Horn, R., Adriaans, P., and van den Berg, D. (2023). The partition problem, and how the distribution of input bits affects the solving process (submitted).

Serdyukov, A. (1978). O nekotorykh ekstremal'nykh obkhodakh v grafakh. *Upravlyayemyye sistemy*, 17:76–79. Original article in Russian is here: http://nas1.math.nsc.ru/aim/journals/us/us17/us17_007.pdf.

Sleegers, J., Olij, R., van Horn, G., and van den Berg, D. (2020). Where the really hard problems aren't. *Operations Research Perspectives*, 7:100160.

Solberg, H. A. and Gaustad, T. (2022). International sport broadcasting: A comparison of european soccer leagues and the major north american team sports. *Sport Broadcasting for Managers*, pages 84–102.

Thielen, C. and Westphal, S. (2010). Approximating the traveling tournament problem with maximum tour length 2. In *Algorithms and Computation: 21st International Symposium, ISAAC 2010, Jeju, Korea, December 15-17, 2010, Proceedings, Part II 21*, pages 303–314. Springer.

Thielen, C. and Westphal, S. (2011). Complexity of the traveling tournament problem. *Theoretical Computer Science*, 412(4-5):345–351.

Thielen, C. and Westphal, S. (2012). Approximation algorithms for ttp (2). *Mathematical Methods of Operations Research*, 76(1):1–20.

Uthus, D. C., Riddle, P. J., and Guesgen, H. W. (2009). An ant colony optimization approach to the traveling tournament problem. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 81–88.

van den Berg, D. and Adriaans, P. (2021). Subset sum and the distribution of information. In *IJCCI*, pages 134–140.

van Eck, O. and van den Berg, D. (2023). Quantifying instance hardness of protein folding within the hp-model (accepted, publication pending). In *IEEE CIBCB 2023*.

Verduin, K., Weise, T., and van den Berg, D. (2023). Why is the traveling tournament problem not solved with genetic algorithms?

Wagner, M. (2016). Stealing items more efficiently with ants: a swarm intelligence approach to the travelling thief problem. In *Swarm Intelligence: 10th International Conference, ANTS 2016, Brussels, Belgium, September 7-9, 2016, Proceedings 10*, pages 273–281. Springer.

Wagner, M., Lindauer, M., Mısır, M., Nallaperuma, S., and Hutter, F. (2018). A case study of algorithm selection for the traveling thief problem. *Journal of Heuristics*, 24:295–320.

Weise, T., Li, X., Chen, Y., and Wu, Z. (2021). Solving job shop scheduling problems without using a bias for good solutions. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1459–1466.

Westphal, S. and Noparlik, K. (2014). A 5.875-approximation for the traveling tournament problem. *Annals of Operations Research*, 218:347–360.

Wu, J., Polyakovskiy, S., Wagner, M., and Neumann, F. (2018). Evolutionary computation plus dynamic programming for the bi-objective travelling thief problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 777–784.

Wu, J., Wagner, M., Polyakovskiy, S., and Neumann, F. (2017). Exact approaches for the travelling thief problem. In *Simulated Evolution and Learning: 11th International Conference, SEAL 2017, Shenzhen, China, November 10–13, 2017, Proceedings 11*, pages 110–121. Springer.

Xiao, M. and Kou, S. (2016). An improved approximation algorithm for the traveling tournament problem with maximum trip length two. In *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Yafrani, M. E., Chand, S., Neumann, A., Ahiod, B., and Wagner, M. (2017). Multi-objectiveness in the single-objective traveling thief problem. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 107–108.

Yafrani, M. E., Martins, M. S., Krari, M. E., Wagner, M., Delgado, M. R., Ahiod, B., and Lüders, R. (2018). A fitness landscape analysis of the travelling thief problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 277–284.

Yafrani, M. E., Scoczynski, M., Delgado, M. R., Lüders, R., Nielsen, P., and Wagner, M. (2022). On the fitness landscapes of interdependency models in the travelling thief problem. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 188–191.

Yamaguchi, D., Imahori, S., Miyashiro, R., and Matsui, T. (2011). An improved approximation algorithm for the traveling tournament problem. *Algorithmica*, 61(4):1077–1091.

Zhao, J. and Xiao, M. (2021). The traveling tournament problem with maximum tour length two: A practical algorithm with an improved approximation bound. In *IJCAI*, pages 4206–4212.

Zhao, J., Xiao, M., and Xu, C. (2022). Improved approximation algorithms for the traveling tournament problem. In *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.