A Study of Multi-objectiveness in the Travelling Thief Problem

Mohamed El Yafrani
m.elyafrani@gmail.com
Local supervisor: Belaïd Ahiod, Mohammed V University in Rabat
Host supervisor: Markus Wagner, The University of Adelaide
Collaborating PhD students:
Shelvin Chand, UNSW, Canberra
Aneta Neumann, The University of Adelaide

***Abstract*—Multi-component problems are optimization problems that are composed of multiple interacting sub-problems. The components are mostly NP-complete problems which makes solving such problems quite challenging. The concept of interdependent components is very important in real-world problems, mainly in industrial applications such as the optimization of supply chains. The motivation of this work is to investigate whether it can be better to consider multiple objectives when dealing with multiple interdependent components. Therefore, the Travelling Thief Problem, a relatively new benchmark problem, is investigated as a bi-objective problem. In our experimental study, an NSGA-II adaptation for the bi-objective model is compared to two of the best known algorithms for the original single-objective problem. The results show that the proposed EMOA does not only generate a range of solutions, but is also competitive with the state-of-the-art single-objective algorithms.**

***Index Terms*—Interdependence; Multi-component problems; Evolutionary Multi-objective Optimization; Travelling Thief Problem**

I. Introduction

In the CEC 2013 conference, Bonyadi et al. proposed a benchmark problem called The Travelling Thief Problem (TTP) [2, 14]. TTP combines the Travelling Salesman Problem (TSP) and the Knapsack Problem (KP) in order to simulates problems that are composed of multiple connected sub-problems, known in literature as multi-component problems. Since each sub-problem is an NP-complete combinatorial optimization problem, each component naturally has an objective function to optimize.

In the original paper, the authors proposed two versions of the problem: a mono-objective TTP (TTP1) and a bi-objective TTP (TTP2). However, all published papers known to us are only investigating the mono-objective version.

The first attempt to solve the problem efficiently was made by Mei et al. [13] using a Memetic Algorithm. In the paper, the authors also proposed speedup and approximation techniques in order to solve large instances efficiently in under 10 minutes.

Faulkner et al. [10] proposed multiple routines which were combined in different manners in order to design more sophisticated heuristics. The best performing heuristic is named *S5*, a simple heuristic that has two stages. First, the Lin-Kernighan heuristic [11] was used to generate a tour independently from the KP part. Second, after fixing the tour, an iterative heuristic named *PackIterative* was used to create a picking plan optimized accordingly to the fixed tour. This simple process was shown to be very efficient as *S5* was able to beat *MATLS* and the other proposed heuristics for most TTP instances.

Since most proposed heuristics use Lin-Kernighan to initialize the tour, which is quite biased towards the TSP part, Wagner [16] took a different path by investigating longer tours. The author adapted a *Max-Min Ant System*, initially designed for the TSP, to construct tours that are likely to be longer, then uses state-of-the-art heuristics to improve the picking plan accordingly. This approach is shown to be very efficient for small TTP instances.

El Yafrani and Ahiod [9] recently proposed two heuristics and carried on an empirical study to compare population-based and single solution heuristics. The first proposed heuristic is called *MA2B*, a memetic algorithm using 2-opt and bit-flip local searches. The second is a combination of a 2-opt local search and a simulated annealing based heuristic for efficient packing, named *CS2SA*. The two proposed heuristics were shown to be very competitive to *MATLS* and *S5*. *MA2B* performs particularly well on small instances, while *CS2SA* was more efficient on large ones.

In a tentative to propose a more realistic version of TTP, Chand and Wagner [3] proposed a version that allows multiple thieves, which is quite close to the Vehicle Routing Problem. The paper also proposes a set of search heuristics for this version of the problem.

In [8], the authors focus on designing a TTP specific neighborhood instead of using a sequential structure as in most heuristics. The results show that this approach was competitive to *EA* and *RLS* on different small instances.

The motivation of this work is to investigate multi-component problems as multi-objective ones by taking the TTP as a benchmark problem. In this report, we are investigating the TTP as a bi-objective problem by considering traveling time and profit as overall objectives.

The investigation of this bi-objective model allowed us to see that the best known TTP solutions can be found in the knee region of the Pareto front. The EMOA was even able to compete with two of the best algorithms for the TTP and find

better solutions for the single objective model implicitly.

Using a multi-objective model is very beneficial in real-world applications since it allows more freedom for the decision makers. In our case, not only we were able to find a trade-off of solutions for TTP, but our EMOA was competitive even for the single-objective model on a sub-set of small TTP instances.

The rest of this report is organized as follows: In Section II, the original TTP is briefly revisited. Our bi-objective TTP is defined in Section III. Section IV describes the proposed EMOA and its components. Our strategies for tuning the EMOA and experimental results are reported in Section V. Finally, conclusions are drawn in Section VI.

II. The Travelling Thief Problem

In the TTP, as redefined by Polyakovskiy et al. [14], we are given a set of n cities and a set of m items distributed among the n cities. Each item k is defined by its profit p_k and weight w_k. A thief must visit all cities exactly once, steal some items on the road, and return to the first city.

In addition, we consider the following parameters and constraints:

- The total weight of the collected items must not exceed a specified capacity W.
- The knapsack is rented, and that the renting rate per time unit is noted R.
- We consider that the thief has a maximum and minimum velocities denoted v_{max} and v_{min} respectively.
- Each item is available in only one city. We note $A_i \in \{1, \ldots, n\}$ the availability vector, such as A_i contains the reference to the city that contains the item i.
- To make the sub-problems mutually dependent, the speed of the thief changes according to the knapsack weight. Therefore, the thief's velocity at city x is defined in Equation 1.

$$v_x = v_{max} - C \times w_x \tag{1}$$

where $C = (v_{max} - v_{min})/W$ is a constant value, and w_x the weight of the knapsack at city x.

We note $g(z)$ the total value of all collected items and $f(x, z)$ the total travel time which are defined in Equations 2 and 3 respectively.

$$g(z) = \sum_m p_m \times z_m \tag{2}$$

$$\text{subject to} \sum_m w_m \times z_m \leq W$$

$$f(x, z) = \sum_{i=1}^{n-1} t_{x_i, x_{i+1}} + t_{x_n, x_1} \tag{3}$$

The overall objective is to maximize the travel gain, as defined in Equation 4, by finding the best tour and picking plan.

$$G(x, z) = g(z) - R \times f(x, z) \tag{4}$$

We will also refer to the gain function as the *TTP score*.

In our implementations, a TTP solution is naturally coded in two parts. The first is the tour $x = (x_1, \ldots, x_n)$, a vector containing the ordered list of cities. The second is the picking plan $z = (z_1, \ldots, z_m)$, a binary vector representing the states of items (0 for packed, and 1 for unpacked).

III. Multi-objective TTP: the proposed model

In this work, we propose to investigate the TTP as a bi-objective problem, which we will simply refer to as MO-TTP. Therefore, we consider the profit (g) and the travelling time (f) as the objectives as shown in Equation 5.

$$\begin{aligned} &\text{maximize } g(z) \\ &\text{minimize } f(x, z) \end{aligned} \tag{5}$$

Note that we keep all the other aspects of the single-objective model, except for the renting rate. In fact, since we are dealing with travelling time and profit separately, there is no benefit on using R which basically acts as a weight in the TTP's formula.

In addition, The two objectives of MO-TTP are conflicting due to the interdependence between the two sub-problems. This interdependence is guaranteed by the changing speed constraint in Equation 1.

This model is different from *TTP2* which was proposed in [2]. In fact, we do not consider the additional constraint which supposes that the value of the items decreases with time.

IV. Solving MO-TTP

A. Overall algorithm

Our EMOA is based on the *NSGA-II* algorithm [5] implemented using jMetal [7]. We define two disruptive operators and two local search heuristics as *NSGA-II* mutations. The *Null Crossover* is used to simply clone selected solutions into the next generation. At each generation, the mutations operators are applied, then the solutions are sorted using the non-dominated sorting to construct multiple fronts. Afterwards, the crowding distance is used to measure the proximity of a given solution to its neighbors. The crowding operator will prioritize the solutions located in a less crowded region. Based on these two operators, the solutions for the next generation are selected. The *NSGA-II* process is repeated until the stopping criterion is met. In our adaptation, we use a runtime limit of 10 minutes as a stopping criterion. In the rest of this report, we will refer to this algorithm as *EMOA-TTP*.

B. Initialization approaches

We use the Lin-Kernighan heuristic to generate the initial tours [11]. As for the picking plans, we use the following initialization strategies.

- **Random picking plan (RPP):** simply generates a random picking plan, such as the total capacity does not exceed the limit W.
- **Greedy picking plan (GPP):** the picking plan is generated using a greedy algorithm that uses a goodness

score to sort items. The score is based on approximations of the benefit of packing/unpacking an item. The reader is referred to [13] for further details about this greedy algorithms.

- **PackIterative (PI):** this strategy is also based on a greedy algorithm. Each item receives a goodness score that depends on its profit, weight, and remoteness from the last city in the tour. The profit and weight are strengthened using an exponent. This process is repeated iteratively and the exponent value is updated at each iteration. Furthermore, since the objective function is very time consuming, the frequency of its use is reduced. This heuristic is explained in more details in [10].

The percentages of using each of these strategies is tuned using the *irace package*, which we explain in Section IV.

C. Mutation operators

In our EMOA, we use the following two disruptive operators.

- **Node insertion:** this operator is applied to the tour. It changes the position of one city, picked at random, in the tour.
- **Random bit-flip:** this operator acts on the picking plan. It goes through all the picking plan bits, flips the current item depending on a probability of $10/m$.

Given that the above operators are used for diversification purposes. We also use the following two hill climber heuristics.

- **2-opt based local search:** a hill climber that uses a neighborhood generated using the 2-opt operator [4]. In addition, the Delaunay triangulation is used to reduce the complexity of the neighborhood [6].
- **Bit-flip based local search:** a hill climber using the bit-flip operator to generate a neighborhood.

The fitness used in both local search heuristics is the TTP score, and the stopping criterion is having no improvement during a complete iteration.

D. Selection

We use a binary tournament selection alongside with the crowding distance to determine which individuals will be passed to the next generation.

In addition, we created a second version of the algorithm in which we introduce another selection operator based on the TTP score. The operator is induced before the crowding distance selection in order to prioritize solutions with higher TTP scores. We will refer to this selection operator as the **Biased Selection**.

V. Experimental study

A. Benchmark Instances

The experiments conducted are performed on a subset of the TTP benchmark instances from [14]. The characteristics of these instances vary widely, and in this work we consider the following problem parameters:

- The number of cities is based on TSP instances from the TSPlib, described in [15].
- For each TTP instance, there are three different types of knapsack problems: uncorrelated, uncorrelated with similar weights, and bounded strongly correlated types.
- For each TSP and KP combination, the number of items per city (item factor, denoted F) is $F \in \{1, 5, 10\}$
- For each TTP configuration, we use 3 different knapsack capacities $C \in \{1, 5, 10\}$. C represents a capacity class.

Furthermore, we use the following representative small sized TTP instance groups to conduct our experiments: *eil51*, *berlin52*, *eil76*, *kroA100*, *kroA150*, *u159*, *a280*, *pr439*, *rat575*, and *rat783*.

B. Parameter tuning

EMOA-TTP has many parameters that need tuning. These parameters are presented in Table I and tuned using the irace package [12]. The irace package is an *R* framework that implements the iterated racing procedure, an extension of the Iterated F-race procedure [1]. Its main purpose is to automatically configure algorithms by finding the most appropriate settings given a set of instances of an optimization problem.

TABLE I: A list of the *EMOA-TTP* paramaters

Notation	Description
p_{ls2}	Probability of applying the 2-opt based local search operator
p_{lsb}	Probability of applying the bit-flip local search operator
p_{mui}	Probability of applying the node insertion operator
p_{mub}	Probability of applying the random bit-flip operator
N	Population size
p_{ir}	Proportion of individuals initialized using RPP
p_{ig}	Proportion of individuals initialized using GPP
p_{ip}	Proportion of individuals initialized using PI
r_{bs}	Biased selection replacement rate

The TTP library proposed by Polyakovskiy et al. [14] is a rich and diverse database. Since the instances are very different in size and type, we investigate multiple tuning strategies. Table II describes all these tuning strategies.

The optimized parameters for each strategy are reported in Table III. It is very difficult to interpret the configurations or to see patterns, as the space is rather high-dimensional.

C. Results and discussion

Figure 1 represents the obtained Pareto front for a various subset of solutions. Note that these results correspond to the *SS* tuning strategy. The figures show that *EMOA-TTP* was able to obtain a set of solutions that represent a tradeoff between time and profit. We can also see that the best solutions regarding the TTP score are concentrated in the knee region of the pareto front. In addition, the solutions obtained using *MA2B* and *S5* are always close to the knee region. This shows that the single objective model is somehow contained in the multi-objective model we investigated. Therefore, the TTP score is a simple scalarization of a bi-objective problem by nature.

TABLE II: The strategies used for tuning *EMOA-TTP*

Notation	Maximum number of experiments	Description
DS	10000	(Diverse Set) Uses the following various set of instances for training: eil51, eil76, kroA100, a280, pr439, rat575, rat783. This strategy considers all the parameters except the biased selection replacement rate (r_{bs}).
DS-BS	10000	(Diverse Set and Biased Selection) Uses the same training set in DS to tune all the parameters.
SS	10000	(Small Set) Uses a small set of instances for training: eil51, berlin52, eil76, kroA100. All the parameters except r_{bs} are optimized.
group*	5000	The training set in this strategy is restricted to one group of instances. For instance, eil51* corresponds to all the eil51 instances in our TTP subset. All the parameters except r_{bs} are optimized.

TABLE III: The obtained elite configurations using the irace package

Strategy	p_{ls2}	p_{lsb}	p_{mui}	p_{mub}	N	p_{ir}	p_{ig}	p_{ip}	r_{bs}
DS	0.1479	0.295	0.5974	0.0708	400	0.8677	0.8593	0.6194	0
DS-BS	0.2045	0.114	0.055	0.7372	340	0.4126	0.5496	0.7318	0.4802
SS	0.1905	0.1667	0.6861	0.5172	400	0.9614	0.5326	0.1839	0
eil51*	0.1047	0.1638	0.6239	0.6966	360	0.488	0.9445	0.333	0
berlin52*	0.1002	0.0479	0.5116	0.4415	380	0.2709	0.8281	0.3713	0
eil76*	0.5654	0.3455	0.3424	0.2019	240	0.3267	0.5483	0.2035	0
kroA100*	0.7363	0.4303	0.9971	0.2343	220	0.3426	0.6117	0.7429	0
u159*	0.2034	0.4191	0.2434	0.391	380	0.5004	0.6395	0.9196	0
a280*	0.1528	0.2695	0.4873	0.6663	260	0.6793	0.0253	0.3386	0

We compared the TTP scores obtained using *EMOA-TTP* with two state-of-the-art single-objective algorithms, namely *MA2B* and *S5*. The results are reported in Tables IV, V, VI, VII, VIII, IX for all the tuning strategies we tested. Note that we only report the best found configuration (elite configuration) according to the irace package. The last two columns correspond to the number of times an algorithm achieved the best score and second best score. The results show that *EMOA*-TTP was implicitly able to obtain good TTP scores. By implicitlty we mean that the TTP score was not part of the bi-objective optimization process. *EMOA-TTP* was able to surpass *S5* on the majority of the instances, and was competitve to *MA2B*.

Our MO-TTP model seems to be an elegant alternative to the single objective one. However, efficient multi-objective optimization is still to this day a challanging task. While our algorithm was able to perform decently on small TTP instances, its performance decreases for larger ones.

The results show that the best TTP scores obtained by *EMOA-TTP* correpond to the strategy of training each group of instances separately (*group**). In fact, *group** was able to obtain the best TTP score 31 times. However, training with small instances (*SS*) was also a good strategy even on larger instances. *SS* was ranked first 28 times and, unline *group**, the training phase is done only once.

DS-BS performed well on the *berlin52** group, but its overall performance is quite mediocre. Another drawback of this strategy is that it does not return a set of solutions. In fact, due to the biased selection, in all the cases we verified the algorithm only returns one solution. *DS* was ranked best only 8 times, which means that using a diverse set of solutions is not a very good strategy.

VI. Conclusion and future directions

In this technical report, we studied the Travelling Thief Problem (TTP), a novel multi-component problem, as a bi-objective problem. First, we proposed a two objectives problem as an alternative model to the current single objective model. Then, we proposed an evolutionary algorithm based on *NSGA-II* named *EMOA-TTP*. Lastly, we carried out an empirical study of our algorithm.

We consider the obtained results important for two main reasons:

- Our EMOA was able to obtain a range of solutions which allows more freedom and flexibilty, which is very beneficial from a decision making perspective.
- *EMOA-TTP* was competitive to state-of-the-art single-objective algorithms, even if the TTP score was not taked into consideration as an objective. This means that the proposed bi-objective model is representative of the standard single-objective TTP.

As a result, we believe that multi-objective optimization provides a better model when dealing with multi-component problem. However, such generalization is not so obvious and certainly needs more investigation.

In the future, further efforts will be made to improve the scalability of *EMOA-TTP*.

Acknowledgment

This research has been mainly supported by IEEE Computational Intelligence Society Graduate Student Research Grant 2016.

This work was supported with supercomputing resources provided by the Phoenix HPC service at the University of Adelaide.

TABLE IV: Results for the eil51 instances

TTP instances	MA2B	S5	EMOA-TTP			
			DS	DS-BS	SS	eil51*
eil51_n50_bounded-strongly-corr_01	4069	3840	3840	3840	**4269**	**4269**
eil51_n250_bounded-strongly-corr_01	10836	10912	10912	10912	**11322**	11004
eil51_n500_bounded-strongly-corr_01	**26624**	25426	25426	25426	25506	25542
eil51_n50_bounded-strongly-corr_05	4180	3925	3925	3925	5007	**5138**
eil51_n250_bounded-strongly-corr_05	**32086**	30585	30585	30585	30585	30585
eil51_n500_bounded-strongly-corr_05	**77605**	76634	76634	76634	76634	76541
eil51_n50_bounded-strongly-corr_10	10095	9613	9613	9613	10755	**10949**
eil51_n250_bounded-strongly-corr_10	34237	33853	33853	33853	**36315**	35824
eil51_n500_bounded-strongly-corr_10	**79322**	76148	76148	76148	75625	75720
eil51_n50_uncorr-similar-weights_01	1447	1238	1238	1238	**1460**	**1460**
eil51_n250_uncorr-similar-weights_01	**5451**	4999	4999	4999	**5451**	**5451**
eil51_n500_uncorr-similar-weights_01	**13116**	12246	12246	12246	13105	13097
eil51_n50_uncorr-similar-weights_05	2113	1373	1373	1373	2018	**2133**
eil51_n250_uncorr-similar-weights_05	**14428**	12902	12902	12902	12902	12902
eil51_n500_uncorr-similar-weights_05	**29679**	27780	27780	27780	27780	27781
eil51_n50_uncorr-similar-weights_10	**5599**	5420	5420	5420	5461	5461
eil51_n250_uncorr-similar-weights_10	**27806**	26950	26950	26950	26950	26950
eil51_n500_uncorr-similar-weights_10	**55408**	52889	52889	52889	52891	52719
eil51_n50_uncorr_01	2788	2193	2193	2193	**2871**	2853
eil51_n250_uncorr_01	11568	9816	10394	10394	11387	**11577**
eil51_n500_uncorr_01	**22811**	19485	19640	19640	19981	22463
eil51_n50_uncorr_05	**4408**	2926	2926	2926	4132	4127
eil51_n250_uncorr_05	**18813**	16387	16386	16386	16508	16547
eil51_n500_uncorr_05	**46899**	45001	45001	45001	44839	45001
eil51_n50_uncorr_10	6682	6130	6130	6130	6785	**6838**
eil51_n250_uncorr_10	**30235**	29543	29543	29543	29543	29429
eil51_n500_uncorr_10	**64748**	62697	62697	62697	63667	63679
Best	16	0	0	0	6	8
Second best	1	7	7	7	14	12

REFERENCES

[1] Birattari, M., Yuan, Z., Balaprakash, P., and Stützle, T. (2010). F-race and iterated f-race: An overview. In *Experimental methods for the analysis of optimization algorithms*, pages 311–336. Springer.

[2] Bonyadi, M. R., Michalewicz, Z., and Barone, L. (2013). The travelling thief problem: the first step in the transition from theoretical problems to realistic problems. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1037–1044. IEEE.

[3] Chand, S. and Wagner, M. (2016). Fast heuristics for the multiple traveling thieves problem. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO'16, Denver, Colorado, USA. ACM.

[4] Croes, G. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812.

[5] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

[6] Delaunay, B. (1934). Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2.

[7] Durillo, J. J. and Nebro, A. J. (2011). jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771.

[8] El Yafrani, M. and Ahiod, B. (2016a). A local Search based Approach for Solving the Travelling Thief Problem. *Applied Soft Computing*.

[9] El Yafrani, M. and Ahiod, B. (2016b). Population-based vs. Single-solution Heuristics for the Travelling Thief Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO'16, pages 317–324, Denver, Colorado, USA. ACM.

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

[11] Lin, S. and Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516.

[12] López-Ibánez, M., Dubois-Lacoste, J., Stützle, T., and Birattari, M. (2011). The irace package, iterated race for automatic algorithm configuration. Technical report, Citeseer.

[13] Mei, Y., Li, X., and Yao, X. (2014). On investigation of interdependence between sub-problems of the travelling thief problem. *Soft Computing*, pages 1–16.

[14] 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 conference on Genetic and evolutionary computation*, pages 477–484. ACM.

[15] Reinelt, G. (1991). Tspliba traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384.

[16] Wagner, M. (2016). Stealing Items More Efficiently with

TABLE V: Results for the berlin52 instances

TTP instances	MA2B	S5	EMOA-TTP			
			DS	DS-BS	SS	berlin52*
berlin52_n51_bounded-strongly-corr_01	4203	4017	4017	4283	**4455**	**4455**
berlin52_n255_bounded-strongly-corr_01	15799	15642	15642	15758	**15813**	15757
berlin52_n510_bounded-strongly-corr_01	**30661**	30214	30214	30268	30388	30245
berlin52_n51_bounded-strongly-corr_05	14010	11295	13016	13918	**14213**	14198
berlin52_n255_bounded-strongly-corr_05	67705	53684	55889	68722	66376	**69626**
berlin52_n510_bounded-strongly-corr_05	**129619**	102905	108227	124614	125439	127158
berlin52_n51_bounded-strongly-corr_10	16642	8229	15289	**16672**	16393	16601
berlin52_n255_bounded-strongly-corr_10	84975	55949	80639	**90925**	85388	87680
berlin52_n510_bounded-strongly-corr_10	167009	117286	161286	**172362**	167049	170345
berlin52_n51_uncorr-similar-weights_01	1585	1097	1607	**1656**	**1656**	**1656**
berlin52_n255_uncorr-similar-weights_01	10718	8872	9151	9315	10683	**10911**
berlin52_n510_uncorr-similar-weights_01	25549	20630	23569	22032	25632	**25686**
berlin52_n51_uncorr-similar-weights_05	**6703**	4037	5522	6672	6670	6670
berlin52_n255_uncorr-similar-weights_05	28024	20484	24517	**29357**	27609	28206
berlin52_n510_uncorr-similar-weights_05	58706	42815	49922	**60115**	52649	55018
berlin52_n51_uncorr-similar-weights_10	8982	7237	8222	**9088**	8824	8987
berlin52_n255_uncorr-similar-weights_10	**46717**	40512	44057	46205	45505	45589
berlin52_n510_uncorr-similar-weights_10	91397	79010	87983	**93916**	87171	87092
berlin52_n51_uncorr_01	3053	2331	2890	2898	**3107**	3102
berlin52_n255_uncorr_01	19625	17731	19174	19439	19996	**20013**
berlin52_n510_uncorr_01	39060	38416	38902	38939	38953	**39109**
berlin52_n51_uncorr_05	6391	4714	5236	6179	6430	**6450**
berlin52_n255_uncorr_05	36254	31229	33361	**36868**	34739	35440
berlin52_n510_uncorr_05	74325	62867	65746	**75046**	67937	69710
berlin52_n51_uncorr_10	9544	7961	8881	9450	9463	**9642**
berlin52_n255_uncorr_10	42238	39385	40350	**43312**	40786	41293
berlin52_n510_uncorr_10	93157	86049	88945	**94658**	89711	89040
Best	4	0	0	12	5	9
Second best	11	0	0	3	4	7

Ants: A Swarm Intelligence Approach to the Travelling Thief Problem. In *Proceedings of the 10th International Conference on Swarm Intelligence, ANTS 2016*, pages 273–281, Brussels, Belgium. Springer International Publishing.

TABLE VI: Results for the eil76 instances

TTP instances	MA2B	S5	EMOA-TTP			
			DS	DS-BS	SS	eil76*
eil76_n75_bounded-strongly-corr_01	3907	3742	3742	3742	3808	**3984**
eil76_n375_bounded-strongly-corr_01	17704	18119	18119	18119	**18178**	18155
eil76_n750_bounded-strongly-corr_01	33377	33458	33458	33458	**33469**	33207
eil76_n75_bounded-strongly-corr_05	**5874**	5516	5516	5516	5563	5563
eil76_n375_bounded-strongly-corr_05	**61851**	61099.9	59410	59410	61111	61247
eil76_n750_bounded-strongly-corr_05	**104557**	102104	102104	102104	102104	101763
eil76_n75_bounded-strongly-corr_10	**11562**	10325.6	11069	10095	11069	10805
eil76_n375_bounded-strongly-corr_10	64867	64016.3	64836	62003	64869	**66472**
eil76_n750_bounded-strongly-corr_10	**109354**	103018	103018	103018	102187	101413
eil76_n75_uncorr-similar-weights_01	1419	1137	1387	1124	**1480**	1425
eil76_n375_uncorr-similar-weights_01	11313	10627.5	11247	10615	**11758**	11718
eil76_n750_uncorr-similar-weights_01	**21802**	20495	20495	20495	21255	21151
eil76_n75_uncorr-similar-weights_05	**4180**	3592	3592	3592	3971	3932
eil76_n375_uncorr-similar-weights_05	**21928**	20099.9	20097	20097	19771	20360
eil76_n750_uncorr-similar-weights_05	**41688**	38601.5	38573	38573	38635	38807
eil76_n75_uncorr-similar-weights_10	**8285**	7811	7945	7811	7945	7945
eil76_n375_uncorr-similar-weights_10	**37939**	36578	36578	36578	37244	37526
eil76_n750_uncorr-similar-weights_10	**79517**	77494	77494	77494	77932	77740
eil76_n75_uncorr_01	5272	4705.8	5070	4618	5402	**5420**
eil76_n375_uncorr_01	**14188**	13573	13832	13832	14002	13944
eil76_n750_uncorr_01	36180	35947	36330	36330	36330	**36340**
eil76_n75_uncorr_05	**6694**	5789	5789	5789	5951	5789
eil76_n375_uncorr_05	**27891**	27094	27094	27094	27094	27094
eil76_n750_uncorr_05	**52541**	50557.2	50417	50417	50417	50417
eil76_n75_uncorr_10	**9460**	8830	9459	8830	9459	9459
eil76_n375_uncorr_10	**46266**	44892.1	44860	44860	44860	44860
eil76_n750_uncorr_10	**87997**	85664	85664	85664	85369	84881
Best	19	0	0	0	4	4
Second best	1	7	9	6	14	11

TABLE VII: Results for the kroA100 instances

TTP instances	MA2B	S5	EMOA-TTP			
			DS	DS-BS	SS	kroA100*
kroA100_n99_bounded-strongly-corr_01	**4487**	4278	4278	4278	4455	4459
kroA100_n495_bounded-strongly-corr_01	**24793**	24794	24794	24794	24829	24794
kroA100_n990_bounded-strongly-corr_01	**48823**	48759	48759	48759	48759	48759
kroA100_n99_bounded-strongly-corr_05	**19932**	17615	17615	17615	19472	19344
kroA100_n495_bounded-strongly-corr_05	**93705**	92581	92581	92581	93642	93111
kroA100_n990_bounded-strongly-corr_05	**186924**	183523	183523	183523	186379	186865
kroA100_n99_bounded-strongly-corr_10	21541	20052	20052	20052	22067	**22085**
kroA100_n495_bounded-strongly-corr_10	**99763**	**99763**	**99763**	**99763**	**99763**	99402
kroA100_n990_bounded-strongly-corr_10	**206770**	206758	206758	206758	206273	205650
kroA100_n99_uncorr-similar-weights_01	1920	1791	1791	1791	2119	**2327**
kroA100_n495_uncorr-similar-weights_01	**13358**	12054	12054	12054	12746	12392
kroA100_n990_uncorr-similar-weights_01	**34096**	31914	31914	31914	33183	32734
kroA100_n99_uncorr-similar-weights_05	**9201**	7905	7905	7905	8861	8624
kroA100_n495_uncorr-similar-weights_05	**41585**	39440	39440	39440	39060	39440
kroA100_n990_uncorr-similar-weights_05	**81712**	80386	80386	80386	80394	79316
kroA100_n99_uncorr-similar-weights_10	**15054**	13880	13880	13880	14421	14388
kroA100_n495_uncorr-similar-weights_10	**69328**	69223	69223	69223	69275	68297
kroA100_n990_uncorr-similar-weights_10	**141066**	140954	140954	140954	140954	139393
kroA100_n99_uncorr_01	**3952**	3876	3876	3876	3942	3942
kroA100_n495_uncorr_01	**20015**	20011	20011	20011	**20015**	**20015**
kroA100_n990_uncorr_01	**41022**	40578	40681	40681	40699	40699
kroA100_n99_uncorr_05	**10082**	9932	9932	9932	10012	10012
kroA100_n495_uncorr_05	**56255**	55625	55625	55625	55649	55649
kroA100_n990_uncorr_05	**103610**	103570	103570	103570	103570	103570
kroA100_n99_uncorr_10	**15539**	14943	14943	14943	14956	14956
kroA100_n495_uncorr_10	**78991**	78888	78888	78888	78326	78981
kroA100_n990_uncorr_10	**155629**	155540	155540	155540	155585	155061
Best	24	1	1	1	3	3
Second best	0	6	6	6	19	12

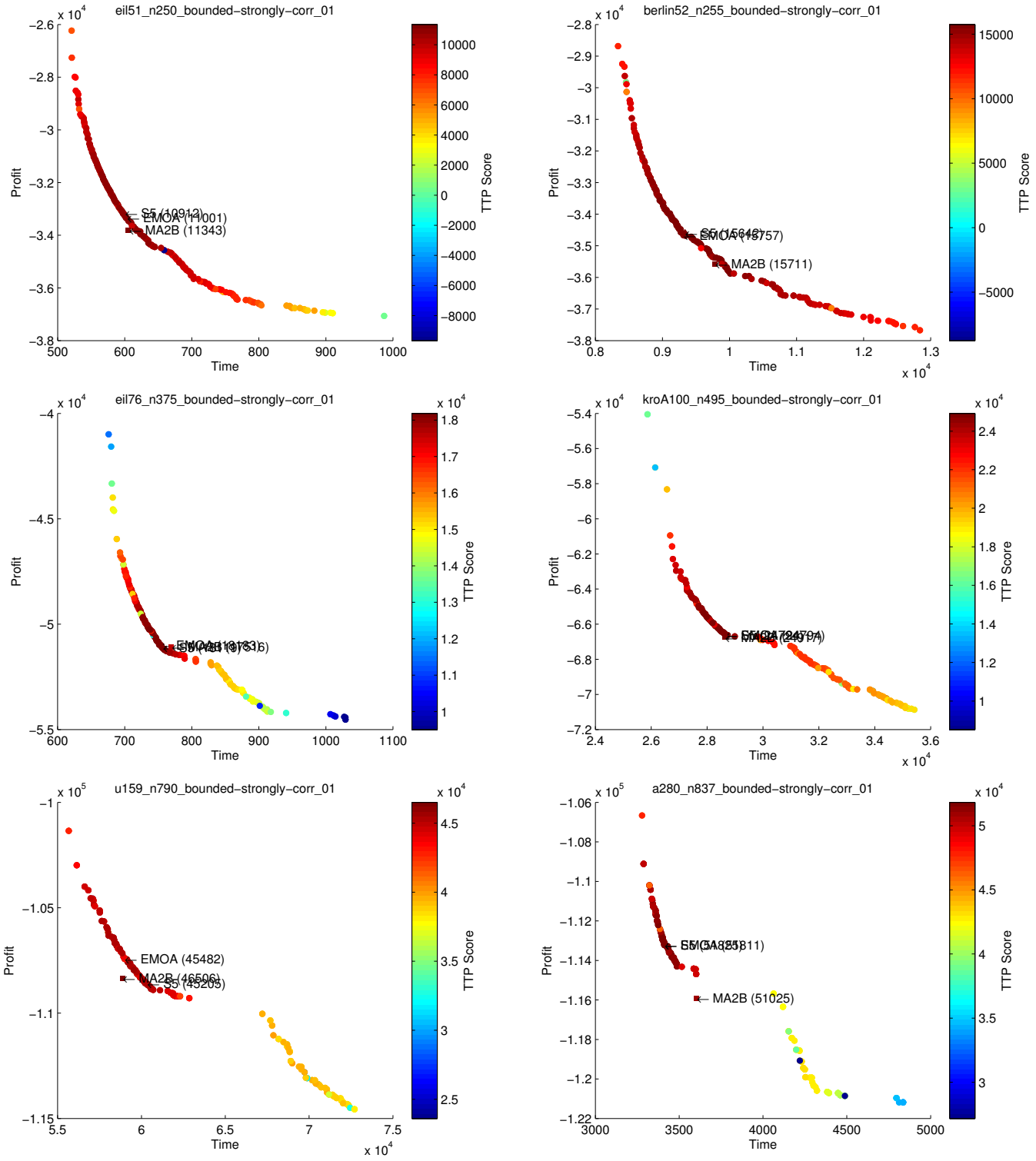

Fig. 1: An illustration of the obtained Pareto front for a subset of TTP instances. The colors represent the TTP score, which is not part of *EMOA-TTP*.

TABLE VIII: Results for the u159 instances

TTP instances	MA2B	S5	EMOA-TTP			
			DS	DS-BS	SS	u159*
u159_n158_bounded-strongly-corr_01	8622	8634	8637	**8672**	8637	**8672**
u159_n790_bounded-strongly-corr_01	**45694**	45205	45462	45502	45493	45496
u159_n1580_bounded-strongly-corr_01	**83962**	83636	83828	83870	83817	83827
u159_n158_bounded-strongly-corr_05	**30712**	29932	30084	30184	30084	30015
u159_n790_bounded-strongly-corr_05	**133387**	133307	133385	133385	133195	133385
u159_n1580_bounded-strongly-corr_05	261372	261261	**261375**	**261375**	**261375**	260876
u159_n158_bounded-strongly-corr_10	38110	37396	**40441**	39772	39502	39290
u159_n790_bounded-strongly-corr_10	**181102**	178796	178887	178887	178887	178740
u159_n1580_bounded-strongly-corr_10	**339560**	339412	339023	339544	339451	338525
u159_n158_uncorr-similar-weights_01	5564	5422	5598	5422	5633	**5691**
u159_n790_uncorr-similar-weights_01	**25510**	24179	24248	24659	24514	24504
u159_n1580_uncorr-similar-weights_01	**49784**	48670	48689	48990	48934	48885
u159_n158_uncorr-similar-weights_05	12205	12205	**13082**	12940	12916	12836
u159_n790_uncorr-similar-weights_05	**58658**	57618	57648	57648	57316	57131
u159_n1580_uncorr-similar-weights_05	**116351**	114465	114536	114536	114536	114536
u159_n158_uncorr-similar-weights_10	23373	22775	23431	**23432**	23431	23175
u159_n790_uncorr-similar-weights_10	**112833**	110062	111556	111569	111286	111152
u159_n1580_uncorr-similar-weights_10	**222792**	217506	220389	222306	218018	220069
u159_n158_uncorr_01	**6212**	5340	5535	5535	5535	5535
u159_n790_uncorr_01	**39359**	38850	39101	39101	39101	39239
u159_n1580_uncorr_01	**79599**	76610	76627	76644	76627	76627
u159_n158_uncorr_05	**20521**	19517	19804	19889	19842	19824
u159_n790_uncorr_05	**87617**	87607	87607	87607	87607	87181
u159_n1580_uncorr_05	184081	**184689**	**184689**	**184689**	**184689**	184379
u159_n158_uncorr_10	25177	24890	**25363**	**25363**	**25363**	25354
u159_n790_uncorr_10	**121013**	119065	119321	119670	119508	119321
u159_n1580_uncorr_10	**244760**	242248	242248	242248	242248	241928
Best	19	1	5	5	3	2
Second best	0	2	8	20	7	4

TABLE IX: Results for the a280 instances

TTP instances	MA2B	S5	EMOA-TTP			
			DS	DS-BS	SS	a280*
a280_n279_bounded-strongly-corr_01	17912	18400	18382	18377	18370	**18433**
a280_n1395_bounded-strongly-corr_01	80882	**83278**	83232	83211	83272	83164
a280_n2790_bounded-strongly-corr_01	152190	156398	156380	156311	156395	**156450**
a280_n279_bounded-strongly-corr_05	55373	55750	**55763**	55695	55696	55740
a280_n1395_bounded-strongly-corr_05	**252903**	250268	250295	249925	250561	251114
a280_n2790_bounded-strongly-corr_05	474104	**478086**	477902	477888	478051	478005
a280_n279_bounded-strongly-corr_10	56226	57073	56772	**57204**	56451	57020
a280_n1395_bounded-strongly-corr_10	292496	303308	303255	303166	**303439**	303379
a280_n2790_bounded-strongly-corr_10	575062	**587116**	586945	585548	586623	586961
a280_n279_uncorr-similar-weights_01	8963	9042	9014	9027	**9055**	9045
a280_n1395_uncorr-similar-weights_01	38724	38727	38699	38677	**38779**	38677
a280_n2790_uncorr-similar-weights_01	78706	79104	79050	78872	**79309**	79137
a280_n279_uncorr-similar-weights_05	**21521**	21395	21403	21330	21403	21414
a280_n1395_uncorr-similar-weights_05	108950	109866	109304	109338	**109923**	109702
a280_n2790_uncorr-similar-weights_05	213732	215570	215395	215429	215460	**215686**
a280_n279_uncorr-similar-weights_10	40268	40749	40761	40718	40717	**40772**
a280_n1395_uncorr-similar-weights_10	194927	**195264**	195155	194936	195122	195182
a280_n2790_uncorr-similar-weights_10	386964	**387594**	386500	387121	387199	387146
a280_n279_uncorr_01	**19246**	18763	18761	18766	19006	18763
a280_n1395_uncorr_01	**68019**	67153	67156	67137	67068	67162
a280_n2790_uncorr_01	**140597**	140532	140519	140507	140410	140543
a280_n279_uncorr_05	**33357**	32883	32886	32844	32843	32845
a280_n1395_uncorr_05	150763	151727	151767	151750	**151785**	151606
a280_n2790_uncorr_05	297309	298654	298203	297358	297905	**298709**
a280_n279_uncorr_10	**42603**	42106	42040	42039	42305	42227
a280_n1395_uncorr_10	208961	209702	209613	209616	**209784**	209731
a280_n2790_uncorr_10	425995	428899	**429002**	428803	428803	428874
Best	7	5	2	1	7	5
Second best	0	9	3	9	5	10