The Travelling Thief Problem: An evolutionary approach

Gabriel Rodrigues, Gonçalo Pereira, José Marcelino, and Sebastian Rehfeldt

Faculty of Science and Technology of the University of Coimbra
University of Coimbra, Portugal
{uc2016211454, uc2008119715, uc2012138018, uc2016181684}@student.uc.pt
https://github.com/goncalonsp/evolutionary-computation-2017/project

Abstract. Real world problems can be really difficult to solve using computer programs. As researchers were able to develop algorithms which efficiently solve specific problems, e.g. the travelling salesman or knapsack problem, the combination of these problems is still a tough task. During this work evolutionary and heuristic algorithms for the combination of these problems, called travelling thief problem, have been developed and evaluated on their capability of solving a complex real-world problem. The results showed that simple evolutionary algorithms alone are not able to find good solutions, but that they can be empowered by problem-specific heuristics, considering the interdependence of the problems. Furthermore, it has been investigated whether it is better to solve such a complex problem at once or to solve the sub-components separately.

Keywords: Evolutionary Algorithms, Genetic Algorithms, Traveling Salesman Problem, Knapsack Problem, Combination, Interdependence, Traveling Thief Problem.

1. Introduction

In the world we live, complexity and lack of knowledge led us to being incapable of solving certain problems. This happens due to the inexistence of an analytical solution to approach the problem, or to unmanageable requirements of processing power. The need of solving these kind of problems is the perfect reason for evolutionary algorithms to exist and have given them importance in the field of solving optimization problems of the real-world.
Nevertheless, the complexity of these problems has been increasing and two new characteristics were added to their complexity, combination and interdependence [1].
The two new concepts refer to the growing complexity of problems, forcing the solution to split the problems in pieces and solving them individually. *Combination* is exactly that, the division of a problem into subproblems. Nevertheless, *interdependence* is a little more important and difficult to address. This concept refers to the relation between the results of a

subproblem and another one, this is, two or more subproblems are related and influence each other.
Since the relevance of these concepts is undeniable, researchers proposed a new benchmark problem for genetic algorithm.[1] The problem proposed is called: *The Traveling Thief*.

The following section presents the travelling thief problem (TTP) formally as well as state-of-the-art solutions, Section 3 is dedicated to the description of our approach and section 4 presents the design, results and discussion of the experiments. Finally, Section 5 states conclusions, new understandings and suggestions of future work.

2. State of the Art

2.1. The Travelling Thief Problem

The Traveling Thief Problem (TTP) is a combination of two popular benchmark problems for genetic algorithms, the Traveling Salesman Problem (TSP) and the Knapsack Problem (KP). The proposal of the TTP was made with the objective of studying the two concepts mentioned in the previous section, combination and interdependence. In this problem, the thief must travel through all cities being able to rob them, with a limited capacity of his knapsack. All cities include one or more items which are represented by a weight and a profit. The thief wants to maximize the profit by visiting each city once and stealing one or more items in each city.
However, the importance of the problem relies on the interdependence of these two problems. This interdependence can be designed differently based on how the problems are tied together. In this study, the first of the two models proposed by Bonyadi et al. [1] has been selected.

2.2. Interdependence models studied for the problem

Both of the models are based on the introduction of a new variable, *time* of a tour. This *time* is directly associated to the length of a given tour, but also to the items picked in each city. Each model presented next, approached this relation between *time* and items differently.

2.2.1. Model TTP1

In this model, two new variables are introduced: velocity and renting of the thief's sack. The model uses as base the physics formula for velocity:

$$v = \frac{d}{t} \quad (1)$$

Then, it was extended to take into consideration the weight of the sack, including its influence into the velocity of the travelling thief:

$$v = v_{max} - w_c \times \frac{(v_{max} - v_{min})}{W} \quad (2)$$

In this new formula, the value for w_c represents the current weight of the thief's sack and *W* its maximum capacity.
The renting of the sack is the variable which, combined with the dynamic velocity and time for finishing a tour, creates the interdependence between these two problems. The outcome is a renting cost:

$$Renting\ Cost(t) = R \times t \quad (3)$$

In formula 3, *R* refers to the constant renting rate of the thief's sack and t (time) can be easily calculated by using the knowledge in (1). The time t depends on both the robbing plan (velocity) and the tour (distance). The goal of this model is to maximize the profit of the thief.

$$Profit = Robbed\ Value(robbing\ plan) - Renting\ Cost(t) \quad (4)$$

Note that *v* is dynamic and changes each time the thief picks a new item. Therefore, the time until the end of the tour is also modified.

2.2.2. Model TTP2

The second model proposed in [1] uses the same definition for velocity (2), therefore, time is also calculated in the same way as in TTP1.
The second variable added to this model is called: *dropping value.* This variable is the one that creates the interdependence between both TSP and KP by associating the value of a specific item to the travel time, this is, the total time the item is carried.

$$Dropping\ value(Item) = item\ value \times Dropping\ rate^{\frac{Travel\ time\ for\ item}{C}} \quad (5)$$

In formula (5), *Dropping rate* is a constant representing the drop of the value of any item and C a constant that controls how much the *Dropping rate* is going to be applied for any item. With this model the objective is to minimize the total time of the trip and to maximize the value of all robbed items after the trip had finished, this is, after applying the *Dropping value*.

2.2.3. Other models

Other models for the design of interdependence for the TTP could rely on other variables like thief's stamina, item's difficulty to be robbed, different values for items according to the city it is found or others.

2.3. Existing approaches

2.3.1. Heuristic approaches

The Travelling Thief Problem is the combination of two NP-Complete problems, therefore, some researchers preferred to focus on studying heuristics to solve deterministically one of the problems, and to build an evolutionary algorithm to evolve the other one.
Since TSP and KP are well known problems in the field, several studies of heuristics have been done to solve them. For the TSP, several heuristics are based on greedy algorithms that select the closest cities with the addition of local search to check if swapping two paths result in a shorter tour or not as in [2,3]. Also, other approaches as the ones presented in [4] use the short tours found by the algorithms mentioned recently and add specific information about the KP knowledge (items that can be found in cities and their information). These latter mentioned algorithms consider the interdependence between the two problems and perform better to the overall solution for TTP. Heuristics for solving the KP are also found, similarly as with the TSP, better solutions are found when considering the interdependence of the problems as in [4].

2.3.1 Co-evolution approaches

Other studies focussed on implementing evolutionary algorithms to tackle interdependence directly. They are based on the idea of exploring a bigger space of solutions where non-optimal combinations of solutions for both problems will be found. This is relevant, since as mentioned in [1], the best solutions for TTP are derived from two non-optimal solutions for TSP and KP.
However, different approaches were taken. In [5] *Co-evolution Collaboration* algorithms and memetic algorithms are developed to study the importance of considering the interdependence to solve the problem. In [6] a *CoSolver* solution is also studied for the problem, where two separated populations are evolved and mixed together for evaluation.
Nevertheless, as studied by [4], the solving of TTP is very hard since very good initial populations are needed as well as good iterated search that is able to escape local minima. Also, big instances create even bigger challenges since the space of solutions grow very fast and the characteristics of the instance are very important for solving it effectively.

3. Methodology and Algorithms

During this work, different evolutionary and heuristic algorithms have been implemented and compared. This section is used to explain on which problem instances we tested our algorithms and gives an idea of how we evaluated our results. Furthermore this part explains how the problem instances can be represented and how they are evolved by our algorithms.

3.1. Model and instance selection

To make the results comparable with other solutions, we decided to follow the TTP1 model which maximizes the overall profit minus the product of renting rate and the time to finish the tour (4). However, as it is shown in [1], both models would be good to study the interdependence since the optimal values are not necessarily found by combining two optimal solutions for both problems, but by combining two eventually non-optimal solutions.

The problem instances were chosen from a competition from 2015 which focussed directly on finding good solutions for the TTP problem.[1] Due to time constraints, we only focussed on instances with 280 cities and 1,5 and 10 items per city. The project also provides functions in Java, Matlab and C# which calculate the objective value of a TTP solution. Furthermore, the so far best-known solutions are given for the problem instances which makes it easily to compare results of new approaches.
During this work, we only used code which was provided by Ernesto Costa in the Evolutionary Computation course at the University of Coimbra. We still adapted heuristics developed by different researchers and came up with additional ideas and heuristics.

3.2. Algorithms overview

There are two main ideas how to solve the TTP problem. As both problems are really well-known and have been studied exhaustively. The first idea is based on solving the two problems separately and maybe iterate the process until convergence or satisfying results is found. Second, the whole problem could be solved at once by finding a representation and fitness function for the complete problem.
The first idea has the advantage of reusing existing approaches which have been proven to be successful. Furthermore, the search space for the complete problem is enormous and is full of local optima which makes it even harder to find a global optimum. Anyway, the first idea lacks the interdependence of the problems and probably is not able to find the best solution as long as they are not interconnected a smart way.
To compare these approaches, a separated and a combined approach have been implemented. The separated approach first solves the TSP problem using an evolutionary algorithm. The top k distinct tours are then used to compute a packing plan based on a deterministic heuristic algorithm. The top k have been chosen because slightly longer tours can be better for the overall problem. Additionally, an evolutionary algorithm using a tuple of a TSP tour and a packing plan as representation was implemented to solve the problem at once. These approaches are explained in more detail after mentioning the necessary heuristic in advance.

3.3. Constructive heuristics

Due to the extremely huge search space which is given by the combination of two NP-complete problems, heuristics became necessary for helping the evolution converging towards good solutions.

3.3.1. TSP heuristics

As mentioned in various papers [3,4,6,7], the Chained Lin-Kerningham heuristic can be used to efficiently find good initial TSP tours. We found pre-calculated TSP tours on a website dedicated to the TTP problem, e.g. in the code for the paper [4].[2]

---

[1] http://cs.adelaide.edu.au/~optlog/CEC2015Comp/

[2] https://sites.google.com/site/mohammadrezabonyadi/standarddatabases/travelling-thief-problem-data-bases-and-raw-results

Furthermore, we implemented our own and simple heuristic for creating short initial tours. This heuristic is based on the idea that you will end up with short tours if you only move to cities which are close to your current city. To keep the performance on a high level, an helper array was created during the reading process of the instance. It contains a sorted list of cities for each city where the closest city is always in the beginning. An individual can then easily be created by starting on a random city and then choose the first city from the helper array which has not been visited yet. The overall population is created by building individuals for a certain amount of distinct random starting cities.
As shorter tours are not always better regarding the final solution, another heuristic has been implemented which focusses more on the items' locations over the cities. The idea under this heuristic is that it can be beneficial to place cities containing more valuable items towards the end of the tour. The intuition is that the items will be carried for shorter time, resulting in a higher velocity at the beginning of the travel and therefore, lower renting cost. The positioning of cities is based on a probability distribution which considers a profit/weight ratio of the items in the cities. However, it is inversed, giving a lower probability to valuable cities, and as a consequence, adding them last, this is, towards the end of the tour. The population is created randomly using the mentioned probability distribution.
As the last two heuristics are used to create initial populations in different ways, it might be beneficial to combine them. An approach using both heuristics was implemented as well. It creates a part of the initial population using the distance heuristic and the rest using the value heuristic. When applying this approach, it seemed that the results were dominated by one heuristic as the distance heuristic created much shorter initial routes. Anyway we consider that a heuristic which directly combines both approaches would be superior.

3.3.2. Solving KP heuristically

As suggested by Polyakovskiy et al. [4], a constructive heuristic could be used to create a packing plan for a fixed TSP tour. In this work, their idea was used to create a packing plan in the separated approach. But it is also plausible to use this heuristic to generate initial packing plans inside the combined algorithm.
The basic idea behind this heuristic is to create scores for all items which indicate how useful they are and then select the ones with the highest scores. The score of an item is the profit the thief would achieve if only this item would be picked during the tour. It is computed by the value of a specific item minus the product of renting ratio and the time from picking the item until finishing the tour.
Unfortunately, not all items are really advantageous and could add a negative value to the objective value. To tackle this problem, Polyakovskiy et al. [4] implemented a fitness value which is given below,

$$fitness\ value\ =\ R \times t' + (p - R \times t_{item}) \quad (6)$$

where *t'* is the total travelling time without any item and t_{item} the total travelling time when only this item would be picked. *R* is the renting ratio and *p* is the profit of the item.
This score can be seen as the improvement of a solution when picking the item and should be positive for advantageous items.

After sorting the items, items with a positive fitness value are picked if they fit inside the knapsack until the knapsack is full.

3.4. Genetic Algorithm for the travelling salesman problem

For solving the TSP problem using a genetic approach, two different approaches were implemented and compared. First, a representation based on random key values was used and secondly, due to the long run time of the first approach, a more direct representation (permutations) was developed.
The random key representation consists of using genotypes with random values in the range [0,1]. The phenotype is obtained by using the indices which would sort the random keys in descending order. This representation was introduced by Bean [8] in 1994 and proven to be successful when the relational ordering of genes is important. [8,9] Another advantage is the easy application of standard genetic operators. Unfortunately, this approach demands a sorting of the genotype for each phenotype or fitness evaluation. Especially for problems with long chromosomes, which is the case in the TTP problem with many cities, this sorting can heavily slow down the computation.
The direct representation was done using permutations. Each genotype is an array of integer values each one mapping to the city index and the phenotype derivation as the name implies is simply a copy of the genotype.
The fitness function for both approaches was to simply compute the distance of the tour represented by the phenotype. More sophisticated functions which already consider the item locations are plausible, but left open to future work. We decided to not implement this as we wanted to focus more on the combined approach.
For crossover operators we used two point crossover (random keys) and cycle crossover (permutations) and for mutation operators gaussian float (random keys) and permutation of pairs (permutations) of genes.
For both approaches, a tournament selection was used for parent selection and elitism for the survivors selection.
Finally, the initial population generation was target of much effort with the objective of integrating domain knowledge about the problem to obtain a faster convergence to what is expected as a good area of the search space. For the random key representation only a random generation was employed but for the direct representation 4 different methods were developed: a random generation, a generation based on a value or distance heuristic and finally a mixed heuristic which uses both the value and distance metrics.

3.5. Genetic Algorithm for the knapsack problem

Again, similar to the TSP, a genetic approach was developed for the KP sub-problem. In this case a binary representation was used where the gene is true if it is picked into the knapsack. The phenotype conversion is a matter of seeing which genes were true and using its index as the item index.
To assess the fitness, the total profit from the picked items was summed if the total weight of the knapsack didn’t surpassed the maximum allowed, when it did the value of 0 was used.

For crossover operator we used uniform crossover and for mutation operator a binary flip of genes. As in TSP, a tournament selection was used for parent selection and elitism for the survivors selection.
A random boolean generation was used to generate the initial population. In order to fix invalid individuals (that violated the knapsack weight constraint) items were removed randomly until each individual was valid.

3.6. Genetic Algorithm for TTP

Using the knowledge obtained by solving the TSP and KP sub-problems separately a genetic approach was then developed for the TTP problem using a Co-evolution method that evolves both problems simultaneously. This approach, is similar to the *Co-evolution Collaboration* presented in [5]. Using this method, each individual is composed of two individuals, one for each sub problem, and the fitness is assessed for the two problems in conjunction using the model specified in section 2.2.1.
Given this, the TTP representation is a tuple of the TSP and KP representations, the operators are the ones used for each subproblem and as in the TSP and KP. Similar to TSP and KP, tournament selection was used for parent selection and elitism for the survivors selection.
The initial population for each sub-representation was again reused from the sub-problems genetic implementation. On the specific case of the TSP problem only the random and distance heuristic where used and experimented with.

3.7. Parameter selection

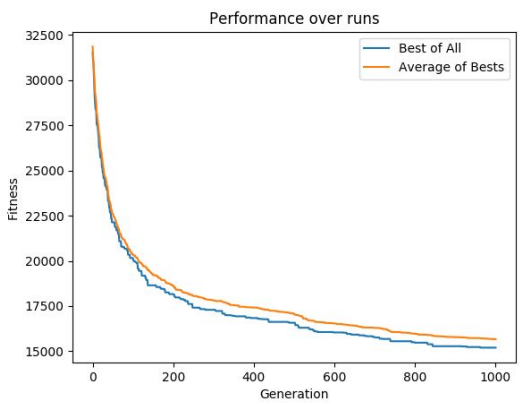
The quality of the algorithms highly depend on the parameters which are used during the evolution. Especially, the probabilities for crossover and mutation seemed to have a big impact on the overall quality. Due to the almost unlimited amount of parameter combinations and the long runtime of the algorithms, we have not been able to conduct statistically valid experiments to find the optimal parameters for our solutions. Instead we tried different parameters for a small number of runs and generations and compared the results by checking the plot for the performance over generations and runs and the achieved objective values for the overall problem.

4. Experimental Results

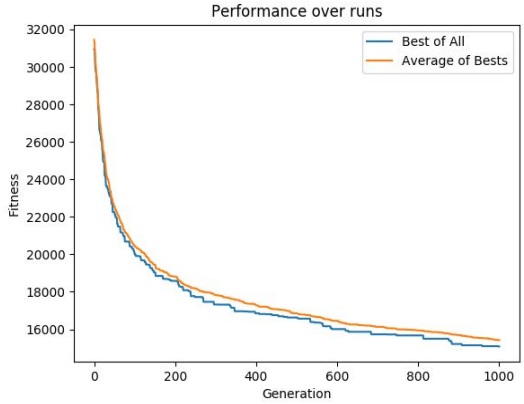
Different experiments have been conducted during this project to compare the different approaches and to justify design decisions. Concretely, experiments have been conducted to compare the two representations for the TSP (4.1), the different heuristics for the initial population (4.2) and to compare the final results of the combined approach, the best separated approach and the best-known results (4.3). The results are only derived from the set of a280 instances. These instances cover already a broad range of problem instances and have a reasonable runtime.

4.1. Comparison of tour representations

This experiment was conducted to compare the two different representations for the TSP EA. Due to the long runtime, we only did 10 runs on the instance with 280 cities and 1 item each and stopped the algorithm after 1000 generations. The results are shown in the figures below.



Results for the direct permutation



Results using random keys

It can be seen that the random key and permutation representation are quite similar and led to similar results of lengths around 16k. Due to the heavy sorting and hence, long runtime, we decided to discard the random key representation and focussed our further work on the representation based on permutations. We are aware that 10 runs are few to draw statistically valid conclusions, but in this case we already expected similar results and rely on our intuition that the results are not too distinct.

4.2. Comparison of initialization heuristics

To evaluate our theory that initialization heuristics could lead to significantly shorter tours, we have run the separated algorithm 10 times with random initialization and the value and distance heuristic initializations. Furthermore, we compared the tour lengths with the length of the linkern tour which is 2593 for the a280 instance. Also the objective values are compared to check if longer tours can be better and that the value heuristic has its' justification. We did 10 runs for all a280 instances. The lengths and objective values can be seen in the tables below.

Instance	Random	Value	Distance	Linkern
280 cities, 1 item	15350	15345	3122	2593
280 cities, 5 items	15586	15226	3123	2593
280 cities, 10 items	15392	14893	3246	2593

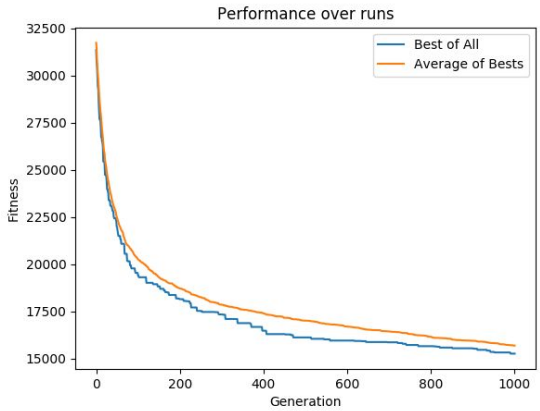
Table comparing the tour lengths

Instance	Random	Value	Distance	Linkern
280 cities, 1 item	-67280	-69024	6496	10619

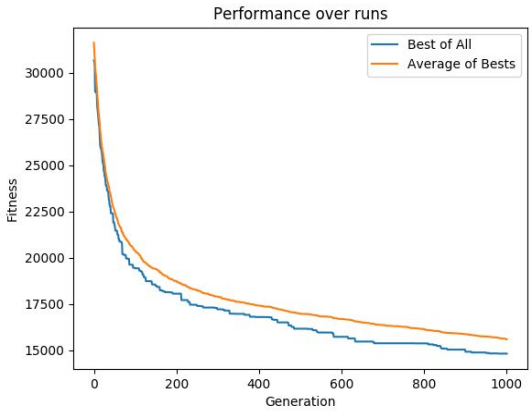
280 cities, 5 items	-1120663	-1069164	-93741	-4651
280 cities, 10 items	-2758977	-2666822	-158018	154519

Table comparing the objective values of the different approaches

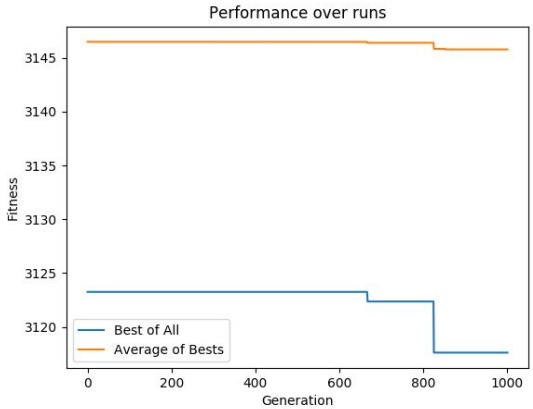
From the tables, it can be seen that the linkern tour is the shortest and best. This also results in the best objective values. It is followed by the algorithm using the distance heuristic which is clearly better than the other two initialization strategies in both length and objective value. The tours are almost 5 times shorter than the others and the objective values are 10 or more times better. The lengths and objective values of using the random initialization and value heuristic based generation are more similar with a slightly plus on the side of the heuristic when regarding the objective value. During our experiments, we figured out that longer tours can be superior to shorter tours. Nevertheless, much longer tours cannot be compensated by a smart packing plan. Especially on instances with higher renting rates, as in the instances with more items per city, long tours are punished hardly. We did not include the mixed heuristic into our results. These were really close to the ones of the distance heuristic as this one dominates the value heuristic. Plots for the performance of the TSP EA over the 10 runs for the three algorithms are given below.



Random initialization



Value heuristic initialization

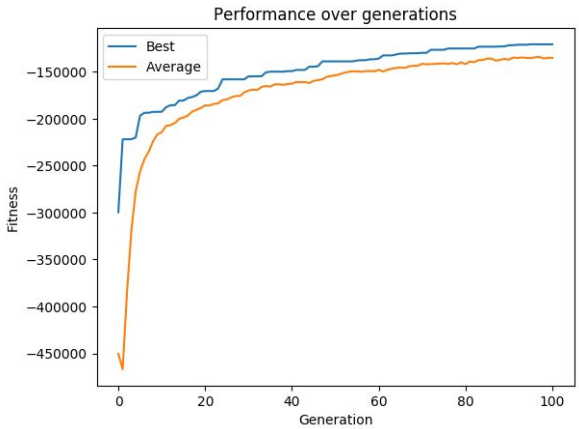


Distance heuristic initialization

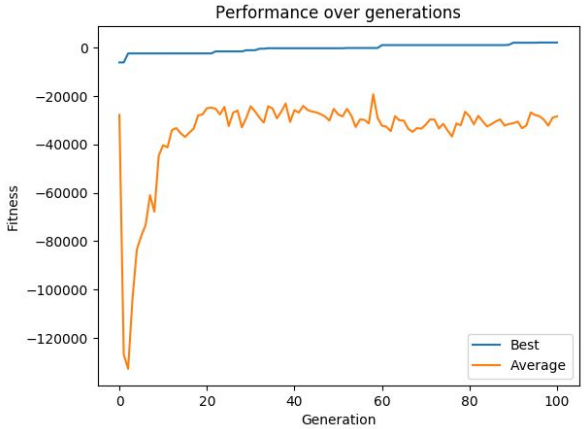
The plots visualize that the first algorithms evolve to much better solutions over the generations. It can also be seen that they still improve after 1000 generations and might get

better. Anyway, their evolution is too slow and result in poor results in comparison to the last approach. Here the best solutions is almost completely given by the initial solution. Even with big probabilities for mutation and crossover, we could barely improve the initial solution. Nevertheless, the algorithm is sometimes lucky and finds better individuals.
Our conclusion from this experiment is that smart heuristics are clearly better than simple evolutionary algorithms. Also the genetic operators do not seem to be perfect as the last algorithm is almost not changing. Furthermore, it can be concluded that better solutions for one problem do not necessarily lead to better overall solutions, but still the results are poor if one problem is solved poorly.

In the case of the TTP initial populations' generation similar experiments were performed in order to test the same hypotheses. Both the random and distance heuristic were tested against the a280 instances and, as can be seen in the graphs below, the distance heuristic does indeed produce noticeable better results since it produces good starting tours leaving the EA to find small improvements in tours and a good packing plan. Given the massive difference in performance no experiment with multiple runs was performed.



Random Initialization of TSP



Distance Heuristic Initialization of TSP

4.3. Comparison of final algorithms

The final results of our algorithms are given by the separated and combined approach using the distance heuristic for initialization. 10 runs and 1000 generations for each of the a280 instance have been conducted for the separated approach, as well as 30 runs and 100 generations for the combined approach. The results were compared to the best-known results and to the results when using linkern tours. The evaluation is based on the objective value of the best individual over all runs. The results can be seen below. Better results for the combined approach could have been achieved using more generations. Unfortunately, the time was not enough to run bigger experiments.

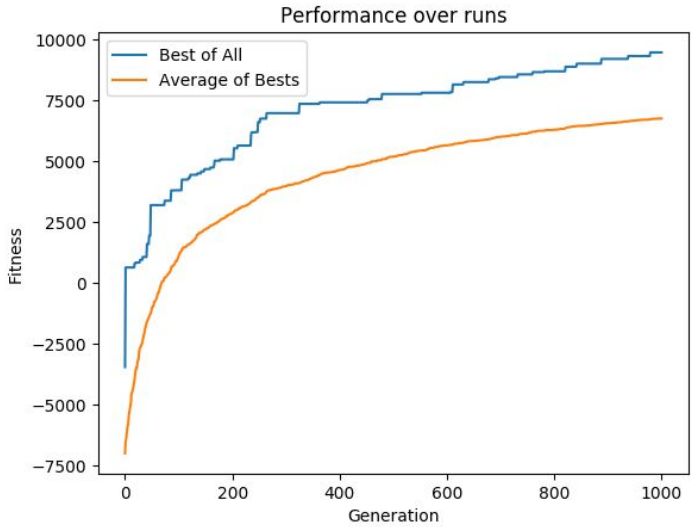
Instance	Separated using dist_heuristic	Combined EA w/ dist_heuristic	Linkern	Best-known[3]
280 cities, 1 item	6496	5035	10619	18049

[3] http://cs.adelaide.edu.au/~optlog/CEC2015Comp/

280 cities, 5 items	-93741	-137277	-4651	109221
280 cities, 10 items	-158018	-148445	154519	428117

Table comparing the final objective values of the different approaches (maximization)

As already discussed in section 4.2, the separated approach comes close to the linkern approach for the instance with just 1 item per city, but gets much worse for bigger problem instances. The huge difference in the objective value there, can be explained by the longer tour length and the higher renting rate for the second and third instance.
The combined approach stays slightly behind the separated approach in terms of the objective value in some cases and ahead in the larger case. This can be explained by the few generations we were able to run it and by the enormous search space it needs to cover. Nevertheless, the combined approach has its' merits as it uses metrics for fitness which directly covers the interdependence of the given problem and is capable of generating good solutions as the following graph shows (although we did not had the time to prove properly).



TTP EA w/ distance heuristic over 1000 generations

5. Conclusion

During this work, different approaches for solving the TTP problem have been introduced. The presented algorithms are mostly evolutionary algorithms and are supported by different heuristics. Special importance was put on the evolutionary algorithm which creates tours for the thief. This algorithm used two different approaches which turned out to be of equal quality regarding the tour lengths. We decided to use a direct permutation approach as this one was superior in terms of run time. As the results of the simple EA were not satisfying, we added different heuristics to shorten the tour lengths. It turned out that even simple heuristics outperform simple EAs using standard genetic operators.
When comparing with other works, we can conclude that the evolutionary algorithms without heuristics are magnitudes worse than the best-known solutions. The use of heuristics helped a lot on improving our results and come close when handling smaller and easier instances.
During this work, we also introduced a combined solver and compared it to our separated approach. We figured out that although and EA with a greater number of generations would

outperform our heuristics in the smaller instance, the time taken to run that solution and perform actual comparisons with all instances would be too great for our timeframe. Using our final parameters gave us poorer results than our heuristics.
Additionally by implementing the TTP EA a great level of knowledge was gained into how much the representation and initial populations vary the output of the algorithm.
It is also noticeable how different instances can be much more difficult to solve than other ones. This might be related to specific behaviors in the instances and its relations with the used heuristics and evaluation. This may mean that there are no easy generic solutions that can find optimal individuals since the space to search is too big and it is unable to escape local minima. Also the heuristics built have to be robust enough to handle different instances successfully. This intuition needs more studying since it is very relevant to real-world problems possessing interdependence, but for now, understanding the specific instance deeply can help us build better heuristics and reaching better results.
Another conclusion consists of the importance of parameter tuning. Subtle changes in areas such as probability of mutation or crossover can radically change the output of the experiment. In future works, more investigations on the parameters would be needed to further improve the results.

5.1 Future Work

As the travelling thief problem is a more recent problem, much research needs to be done to come up with better and more efficient algorithms.
When looking on our results and algorithms, more work would be needed on the parameter selection and statistical analysis of the results. As the results heavily rely on probabilities, more runs would be needed to draw statistically valid solutions and to find better justifications for design decisions and parameter selections.
Furthermore, more heuristics which consider parts of both sub-problems, interdependence would be needed to assist during the evolution and to help creating good initial solutions. Besides heuristics for the initial population, more sophisticated fitness functions could help to come up with better overall solutions. More concretely, a fitness function in the separated TSP EA could also benefit tours where highly valued items are in the end of the tour instead of just relying on the tour distance. This fitness function would probably also lead to more evolution in the case of good initial solutions. More work would also be needed on the genetic operators for this algorithm.
For the TTP EA implementation side, further work could be performed on developing better starting populations for the KP problem and additional experiments with different crossover and mutation operators for TSP and KP. Mainly on the KP part of the fitness function experiments could also be performed in order to achieve better functions and to correctly assess the impact of the knapsack weight restriction on the overall problem.
Additionally the population evolution an additional step could be introduced where individuals from each subproblem could be combined in order to produce better individuals. The main idea would be to also perform a pseudo-crossover between individuals sharing their TSP or KP representation with each other.
Finally, more research needs to be done on the combination of the separate solvers as the top k approach cannot be the optimal solution. We imagine that an iterative approach which solves the problems in a loop could be beneficial to the top k approach. In our combined

approach, we did not use the knapsack heuristic to find good initial packing plans. It would be interesting to know whether this improves the overall solution. A possible way to do so would be to first generate tours using the distance heuristic and then creating packing plans for them directly in the combined TTP EA.

References

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

[2] M. Hashler, K. Hornik. TSP Infraestructure for the Traveling Salesperson Problem. In Journal of Statistical Software, 23(2), 1-21, 2007.

[3] D. Applegate, W. Cook, and A. Rohe. Chained lin-kernighan for large traveling salesman problems. INFORMS Journal on Computing, 15(1):82–92, 2003.

[4] Polyakovskiy, Sergey, et al. "A comprehensive benchmark set and heuristics for the traveling thief problem." Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation. ACM, 2014.

[5] Y. Mei, X. Li and X. Yao. On investigation of interdependence between sub-problems of the Travelling Thief Problem. In Soft Comput 20:157-172, 2016.

[6] M. R. Bonyadi, Z. Michalewicz, M. R. Przybylek, and A. Wierzbicki. Socially inspired algorithms for the travelling thief problem. In Genetic and Evolutionary Computation Conference (GECCO), pages 421{428, New York, NY, USA, 2014. ACM.

[7] Faulkner, Hayden, et al. "Approximate approaches to the traveling thief problem." Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. ACM, 2015.

[8] Bean, J.: Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing 6 (1994) 154–160

[9] Norman, B., Smith, A.: Random keys genetic algorithm with adaptive penalty function for optimization of constrained facility layout problems. In: Proceedings of the Fourth International Conference on Evolutionary Computation, IEEE (1997) 407–411