On the Impact of the Renting Rate for the Unconstrained Nonlinear Knapsack Problem

Junhua Wu Sergey Polyakovskiy Frank Neumann
Optimisation and Logistics, School of Computer Science
The University of Adelaide, Australia

ABSTRACT

Multi-component problems combine several combinatorial optimisation problems that occur frequently in real-word applications such as supply chain management. In order to study the impact of the combination of such problems, the traveling thief problem [4] which combines the traveling salesman problem and the knapsack problem has been introduced. Recently, it has been shown that the non-linear knapsack problem constituting the packing component of the traveling thief problem is already NP-hard without having the capacity constraint imposed. We investigate the renting rate R which is an important parameter in combining the packing profit and the associated traveling costs in this non-linear knapsack problem. Our theoretical and experimental investigations show how the renting rates influence the difficulty of a given problem instance in terms of how items can be excluded by a simple but very effective preprocessing approach. Furthermore, we carry out theoretical and experimental investigations to create instances that are hard to be solved by simple evolutionary algorithms.

Keywords

Traveling thief problem; Non-linear knapsack problem; Transportation; Combinatorial optimisation

1. INTRODUCTION

Metaheuristic approaches have been applied to a wide range of combinatorial optimisation problems [2, 3]. Apart from classical NP-hard combinatorial problems, multi-component problems have gained special interest in recent years [5, 16]. Such problems combine different combinatorial problems into an overall problem and are well motivated by complexities arising in the area of supply chain management.

Considering multi-component problems, the main question is what makes the combination of the underlying problems harder than solving the different problem separately. Recently the traveling thief problem (TTP) [4] has been introduced as a problem which allows to study the interaction of two well-known combinatorial optimisation problems in a systematic way. The TTP combines two of the most prominent combinatorial optimisation problems, namely the traveling salesperson problem (TSP) and the knapsack problem (KP).

Different approaches have been proposed during the last few years for solving the TTP and a benchmark set based on TSP and KP instances has been presented in [13]. Approaches for dealing with the TTP include various metaheuristics such as randomised local search, evolutionary algorithms, and co-evolutionary approaches [7–10]. It has been shown in [7] that good solvers for the TTP instances of the benchmark set can be obtained by choosing a close to optimal TSP tour of the underlying TSP part and deciding on the selection of items by a simple (1+1) EA. This might suggest that the packing problem is easy once the route is fixed.

However, Polyakovskiy and Neumann [14] have shown that the underlying non-linear knapsack problem (NKP) is already NP-hard without imposing the usual capacity constraint for the knapsack. In this paper, we carry out additional studies for NKP. We pay special attention to an important parameter (R) called the renting rate which connects the profit and the cost part of a packing. Based on the input of the other parameters of the problem, such as the profit and weight of the given items, we derive upper and lower bounds on the interval for R where simple preprocessing approaches presented in [14] is not able to reduce the problem instances. Furthermore, we use an evolutionary algorithm to produce instances that allow to remove as few items as possible for a fixed renting rate R. These studies give additional insights into the importance of R and show the range where difficult instances for given algorithms might be found.

Motivated by the success of simple evolutionary algorithms such as the (1+1) EA for the packing component of the TTP, we investigate how difficult instances of NKP for the (1+1) EA look like afterwards. We do this in two ways. First we construct an instance based on the insights from rigorous runtime analysis for the classical knapsack problem where the (1+1) EA fails with a constant probability to obtain an optimal solution. Afterwards, we evolve an instance for NKP by an evolutionary algorithm where the empirical failure rate of the instance is maximised.

The remainder of this paper explains our investigation in detail. In Section 2, we state the nonlinear knapsack problem (NKP) formally. The investigation on the impact of the renting rate is in Section 3. Section 4 illustrates the ap-

GECCO '16, July 20–24, 2016, Denver, Colorado, USA.

DOI: http://dx.doi.org/XXXX.XXXX

proach of building a pre-processor that can efficiently eliminate easy instances. The theory and the evolutionary algorithm (EA) for finding hard NKP instances are elaborated in Section 5. A final conclusion is drawn in Section 6.

2. PROBLEM STATEMENT

We consider the non-linear knapsack problem, which is obtained from the TTP when the route that corresponds to the TSP part of the problem is fixed. The problem has been introduced in [14] and it has been shown that this non-linear knapsack problem is already NP-hard without imposing the usual capacity constraint on the knapsack.

The NKP can be formally defined as the follows. Given $n+1$ cities, each city i, $1 \leq i \leq n$, contains a set of items, denoted by M_i, having m_i items. Therefore the whole set of items M contains $m = \sum_{i=1}^{n} m_i$ items in total.

Each item e_{ik} is attributed to its integer profit p_{ik} and weight w_{ik}, which are bounded as

$$\forall(e_{ik}) : p_{ik} \in [P_L, P_U]$$

$$\forall(e_{ik}) : w_{ik} \in [W_L, W_U]$$

$$P_U > P_L > 0, W_U > W_L > 0$$

Provided a predefined tour, such as $N = (1, 2, ..., n+1)$, a thief is travelling by a vehicle with velocity within a range of $[v_{max}, v_{min}]$. The benefit of collecting and carrying a set of items $S \subseteq M$ is denoted by B_S,

$$B_S = P_S - R \times T_S \tag{1}$$

$$P_S = \sum_{i=1}^{n} \sum_{k=1}^{m_i} p_{ik} x_{ik} \tag{2}$$

$$T_S = \sum_{i=1}^{n} \frac{d_i}{v_{max} - v \sum_{j=1}^{i} \sum_{k=1}^{m_j} w_{jk} x_{jk}} \tag{3}$$

Here P_S represents the total profit of carrying the set of items and T_S is the corresponding total travel time. As the velocity is influenced by the weight of carried items, the total travel time is increased along with carrying more items. Given a renting rate $R \in (0, \infty)$, $R \times T_S$ is the total cost of carrying item set S.

The $x_{ik} \in \{0, 1\}$ indicates whether or not this item is in S. Therefore the item set S can be represented as a decision vector $X = (x_{11}, x_{12}, ..., x_{1k}, x_{21}, ..., x_{mk})$, and the objective of this problem is to maximise the overall benefit B_X. In addition, $v = \frac{v_{max} - v_{min}}{W}$ is the factor that indicates how the carried weight influences the velocity, where W is the capacity of the vehicle. In our investigation, W is always set to be mW_U in order to be unconstrained.

3. IMPACT OF THE RENTING RATE

Observably from equation (1), the renting rate R has significant influence on the overall benefit. If the R is small enough (i.e. $R \to 0$), then $R \times T \to 0$, which implies the total cost of carrying any set of items is ignorable. The optimal solution of the problem therefore is to carry all items. We denote this optimal solution as $X_{OPT} = \{1\}^m$. We assume there is a boundary R_L so that when $R \in (0, R_L]$, $X_{OPT} = \{1\}^m$ always holds, which makes the problem trivial. Similarly, if $R \to \infty$, then the optimal solution is tended not to carry any item, denoted as $X_{OPT} = \{0\}^m$. We also assume that a boundary R_U exists, so that $X_{OPT} = \{0\}^m$ always holds if $R \in [R_U, \infty)$, which makes the problem trivial as well.

Apparently if R_L and R_U exist, hard instances of NKP can only be found when $R \in (R_L, R_U)$. We therefore name (R_L, R_U) as the *non-trivial range* of the *renting rate*. Our work tend to prove that 1) Hard NKP instances cannot be found when its R is out of the non-trivial range; 2) for the instances with their R being in the range, there is an index related to R, named *non-trivial item ratio*, being able to indicate the hardness of the instances individually as well.

3.1 Theoretical Bounds for Non-Trivial Items

The optimal solution $X_{OPT} = \{1\}^m$ being always tenable is equivalent to the situation that the profit of any item $e_{\theta k} \in M$ always covers the cost of its traveling within any set of items $I \subseteq M$. It can be formalised as $\forall e_{\theta k} \forall I : p_{\theta k} \geq R(T_I - T_{I \setminus \{e_{\theta k}\}})$. In order to find the lower boundary R_L, we can define a new problem in which R need to be maximised subject to $X_{OPT} = \{1\}^m$ holding, denoted as:

$$\max R$$

$$\text{s.t. } \forall e_{\theta k} \forall I : p_{\theta k} \geq R(T_I - T_{I \setminus \{e_{\theta k}\}}) \tag{4}$$

$$e_{\theta k} \in I, I \subseteq M$$

As in equation (3) the denominators depend linearly on the weight of collected items and the travel cost depends on the denominators in inverse proportion along each path of two cities, we have $(T_M - T_{M \setminus \{e_{\theta k}\}}) \geq (T_I - T_{I \setminus \{e_{\theta k}\}})$ [14]. Therefore the inequality in (4) can be strained to $p_{\theta k} \geq R(T_M - T_{M \setminus \{e_{\theta k}\}})$, which is equivalent to:

$$R \leq \frac{p_{\theta k}}{T_M - T_{M \setminus \{e_{\theta k}\}}} \tag{5}$$

$$T_M = \sum_{i=\theta}^{n} d_i \frac{1}{v_{max} - v \sum_{j=1}^{i} \sum_{k=1}^{m_j} w_{jk}}$$

$$T_{M \setminus \{e_{\theta k}\}} = \sum_{i=\theta}^{n} d_i \frac{1}{v_{max} - v \sum_{j=1}^{i} \sum_{k=1}^{m_j} w_{jk} + v w_{\theta k}}$$

As the right end of the inequality (5) has its minimum when $\theta = 1$, $p_{\theta k} = P_L$ and all weights is W_U, we can set R_L as:

$$R_L = P_L \Big/ \sum_{i=1}^{n} \frac{d_i v W_U}{(v_{max} - v m W_U)(v_{max} - v(m-1)W_U)} \tag{6}$$

Similar to R_L, if $R \to \infty$, then the optimal solution is not tended to carry any item, denoted as $X_{OPT} = \{0\}^m$, implying that $\forall e_{\theta k} \forall I : p_{\theta k} \leq R(T_I - T_{I \setminus \{e_{\theta k}\}})$. By setting $J = M \setminus I$, we have:

$$\min R$$

$$\text{s.t. } \forall e_{\theta k} \forall J : p_{\theta k} \leq R(T_{J \cup \{e_{\theta k}\}} - T_J) \tag{7}$$

Route#	From	To	Distance	Route#	From	To	Distance
1	1	22	7	26	42	19	9
2	22	8	12	27	19	40	11
3	8	26	7	28	40	41	12
4	26	31	10	29	41	13	9
5	31	28	6	30	13	25	13
6	28	3	9	31	25	14	6
7	3	36	12	32	14	24	11
8	36	35	6	33	24	43	12
9	35	20	7	34	43	7	12
10	20	2	12	35	7	23	6
11	2	29	9	36	23	48	9
12	29	21	7	37	48	6	9
13	21	16	10	38	6	27	9
14	16	50	6	39	27	51	8
15	50	34	6	40	51	46	2
16	34	30	7	41	46	12	7
17	30	9	8	42	12	47	6
18	9	49	6	43	47	18	8
19	49	10	8	44	18	4	8
20	10	39	10	45	4	17	8
21	39	33	14	46	17	37	5
22	33	45	7	47	37	5	11
23	45	15	7	48	5	38	7
24	15	44	6	49	38	11	7
25	44	42	10	50	11	32	6

Table 1: A Tour Generated by Lin-Kernighan TSP Heuristic for the Eil51 Instance of TSPLIB

$$e_{\theta k} \notin J, e_{\theta k} \in M, J \cup \{e_{\theta k}\} \subseteq M$$

Given $(T_{\{e_{\theta k}\}} - T_{\emptyset}) \leq (T_{J\cup\{e_{\theta k}\}} - T_J)$ [14], we set $p_{\theta k} \leq R(T_{\{e_{\theta k}\}} - T_{\emptyset})$, which is equivalent to:

$$R \geq p_{\theta k} \Big/ \sum_{i=\theta}^{n} d_i \left(\frac{1}{v_{max} - v w_{\theta k}} - \frac{1}{v_{max}} \right) \tag{8}$$

We have R_U when $\theta = n$, $p_{\theta k} = P_U$ and $w_{\theta k} = W_L$:

$$R_U = \frac{P_U v_{max}(v_{max} - v W_L)}{v W_L d_n} \tag{9}$$

As an example, we assume that there are 51 cities located identically as in the eil51 instance of TSPLIB [15]. Each except the last city contains 5 items respectively with bounds as $P_L = W_L = 1$ and $P_U = W_U = 1000$. We also set $v_{max} = 1$ and $v_{min} = 0.1$ to be the maximal and minimal velocities of travelling, additionally capacity $W = 250,000$ so that all the items can be contained in any case. Given a tour as shown in Table 1, we can calculate the bounds to be $R_L = 6.85\text{e}{-2}$ and $R_U = 4.63\text{e}{+7}$ according to equations (6) and (9). With being a finite range, it however is still tremendous and it is also not clear whether the instance is easy or not when R is in the range.

We further investigate whether and how the renting rate R influences the hardness of the problem when R is in the non-trivial range. Instead of focusing on the range of R on instances generally, we investigate whether a specific item $e_{\theta k}$ in an instance has a particular non-trivial range similar to the general non-trivial range. If such particular ranges exist for each of the items in an instance, we use the ratio of the number of items with the R in their ranges on the overall number of items as an index. More specifically an item being trivial means it must be either selected or discarded (i.e. compulsory or unprofitable [14]) when R is out of its own non-trivial range $(R^L_{\theta k},\ R^U_{\theta k})$. We name an item with R in its range $(R^L_{\theta k},\ R^U_{\theta k})$ as a *non-trivial item*, if its range exists. Generally we can define *non-trivial item ratio* as:

Definition *Non-trivial Item Ratio* is the ratio of the number of *non-trivial item* within an instance, denoted by

$$\lambda = \frac{No.\ of\ non\text{-}trivial\ items}{No.\ of\ all\ items}$$

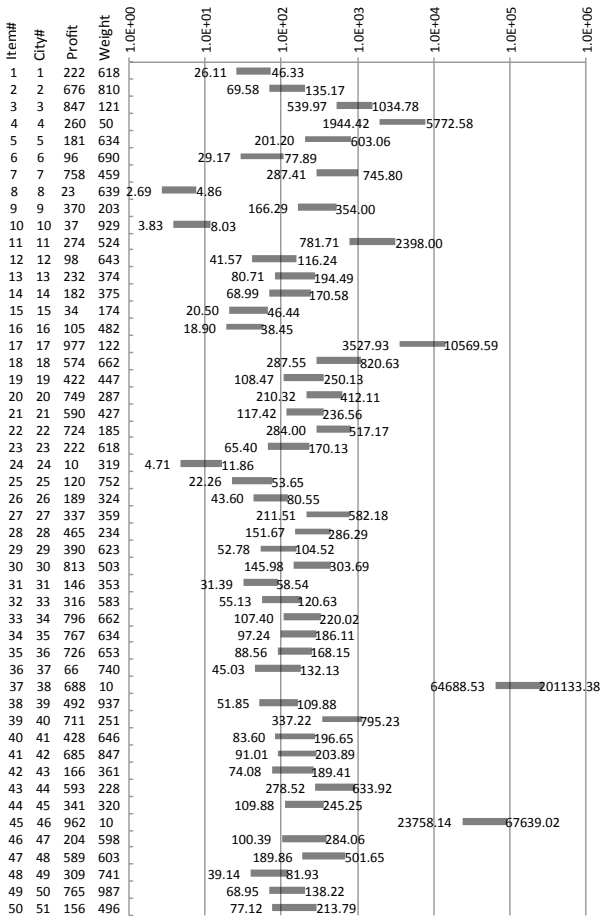


Figure 1: Non-trivial Range of Items with the Instance with 50 Items

In terms of calculating the range $(R^L_{\theta k},\ R^U_{\theta k})$, we set the followed equations according to Proposition 1 and 2 in [14].

$$R^L_{\theta k} = p_{\theta k}/(T_M - T_{M\setminus\{e_{\theta k}\}}) \tag{10}$$

$$R^U_{\theta k} = p_{\theta k}/(T_{\{e_{\theta k}\}} - T_{\emptyset}) \tag{11}$$

Figure 1 shows a particular example of the non-trivial item ranges of an instance according to the equation (10) and (11). This instance contains 50 cities located identically to the eil51 instance in TSPLIB, in which each city except the last one respectively contains one item with arbitrary profit and weight. We set the tour of this instance is identical to the one in Table 1. The ranges of items are represented as the bars in the figure. Here an interesting observation in the figure is that the bars are distributed logarithmically.

In order to obtain *non-trivial item ratio*, we assume there is a predefined renting rate R in the instance shown in Figure 1 and it is a single vertical line in the figure. Obviously this line will only cross some of the bars. Assuming $R = 1.0\text{e}{+2}$ for instance, we have 19 out of 50 items having

n:	50	m:	250
P_L:	1	P_U:	1,000
W_L:	1	W_U:	1,000
v_{max}:	1	v_{min}:	0.1
W:	250,000	v:	3.60e−06

Table 2: The Constants of NKP instances

their bars crossed by R. Those items whose bars are crossed by renting ratio R are non-trivial. The non-trivial item ratio λ therefore equals 0.38 in this case. Similarly, if we set $R = 1.0e+4$, the λ will be 0.02.

With observing the Figure 1 further, we may find that a better *non-trivial item ratio* λ could be obtained if we set R to be around 2.0e+2. However it will be still below 0.5. In fact finding a high λ, 0.8 for example, is not possible in this instance. On the other hand, we would like to have instances with high λ, as intuitively an instance with high λ could be harder than one with low λ. We therefore design a (1+1)-EA to maximising the *non-trivial item ratio* λ.

4. MAXIMISING NON-TRIVIAL ITEM RATIO

In order to obtain NKP instances with high *non-trivial item ratio* λ, we introduce a simple evolutionary algorithm named λEA, in which each weight or profit is mutated in $1/m$ probability. Here m is the number of items. If the mutated instance has a λ better than the one of existing instance, it will be chosen to be the current instance by the selection procedure for the next iteration. The mutation and the selection are continued until the maximal iteration is reached. The detailed pseudocode is listed in Algorithm 1. We then reuse the instance type described in the previous section and the constants of the instance type that are listed in Table 2.

In order to understand whether and how the settings of renting rate R within the *non-trivial range* influence the *non-trivial item ratio*, we run each experiment on a given set of $R \in \{1.0e{-}3, 1.0e{-}2, ..., 1.0e{+}8\}$, which covers the known non-trivial range (6.85e−2, 4.63e+7) calculated in the previous section. We repeat the algorithm 5 times with a unified maximal iteration 100,000 for each given R in the set respectively.

Algorithm 1 λEA for High Non-trivial Item Ratio

```
1: procedure λEA(Maximal Iteration)
2:     Initialisation
3:     repeat
4:         Selection
5:         Mutation
6:     until Reach the maximal iteration
7:     Exit
8: end procedure
9: procedure Initialisation
10:    Initialise an NKP instance
11:    Set a fixed renting rate R
12:    Randomise p_ik ∈ [P_L, P_U] uniformly for each item.
13:    Randomise w_ik ∈ [W_L, W_U] uniformly for each item.
14:    λ = Fitness(initial instance)
15: end procedure
16: procedure Mutation
17:    for each item in the instance do
18:        Randomise b ∈ [0, 1]
19:        if b ≤ 1/m then
20:            Randomise w_ik ∈ [W_L, W_U]
21:        end if
22:        Randomise b ∈ [0, 1]
23:        if b ≤ 1/m then
24:            Randomise p_ik ∈ [P_L, P_U]
25:        end if
26:    end for
27: end procedure
28: procedure Selection
29:    λ' = Fitness(mutated instance)
30:    if λ' > λ then
31:        Choose the mutated instance
32:    end if
33: end procedure
34: function Fitness(instance)
35:    Calculate non-trivial item ratio λ
36:    return λ
37: end function
```

4.1 Results of EA for Non-trivial Item Ratio

The results of λEA are plotted in the Figure 2 as the red line. As a comparison, we also draw a blue line that represents the highest possible λ by purely ramdon instances. It is obtained by generating one million random instances for each given R and picking up the maximal λ calculated among them.

The horizontal axis of Figure 2 represents the range of renting rate. Each error bar on the red line and the corresponding circle on the blue line indicate the R values in the set of $\{1.0e{-}3, 1.0e{-}2, ..., 1.0e{+}8\}$. Additionally, the length of the error bars illustrates the standard deviation of the non-trivial item ratio for each result of λEA. The vertical axis of the figure represents the non-trivial item ratio λ.

According to the Figure 2, both the results of λEA and the random instances contain the values of λ greater than zero when renting rate R is in $\{1.0e0, ..., 1.0e{+}7\}$. In addition, the values of λ obtained from λEA are significantly higher than the ones of randomised instances when R is in $\{1.0e0, ..., 1.0e{+}6\}$, which are also robust according to the standard deviation. By contrast, the results of R in $\{1.0e{-}3, 1.0e{-}2, 1.0e{-}1\}$ and $\{1.0e{+}7, 1.0e{+}8\}$ are close to zero.

If we introduce a threshold of λ, 0.8 for example, for filtering out the settings of R with λ below it, there will be only three values of R left: 1.0e1, 1.0e2 and 1.0e3. This implies that the instances with high λ can only be found when R is in the range that exactly covers the three R values, and this range is much narrower than the *non-trivial range*. Therefore, if it can be confirmed that hard NKP instances can also be only found when R is in this range, we can utilise the threshold as a pre-processor. This pre-processor will filter out the instances with λ below a threshold, which means eliminating easy instances.

On the other hand, the significant increased values of λ by λEA, for instance when $R = 1.0e{+}2$ the λ is increased to be 1 from 0.12, is made by an effect that we called *accumulation* according to our observation. Such an effort accumulates non-trivial ranges of the items to be around the preset value of R via eliminating items with unsuitable profits and

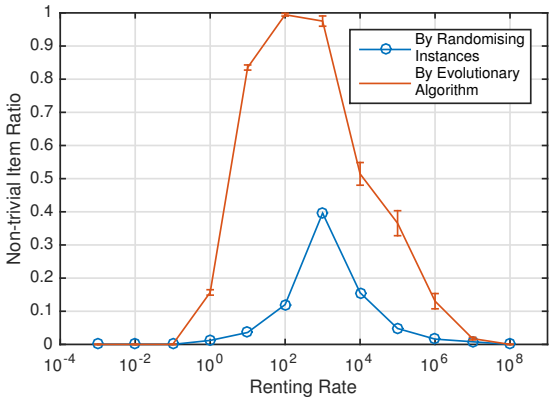


Figure 2: Non-trivial Item Ratio by λEA

weights. The effects of this process can be visualised in Figure 3, in which the Figure 3a is a random instance without accumulation, and the Figure 3b is an instance having been accumulated.

We believe such an effect creates the hard instances of NKP: the more accumulation around the preset value of R, the few items can be easily removed by the pre-processing in [14] and the harder the instance would be, as it would become harder to decide which items should be carried or not.

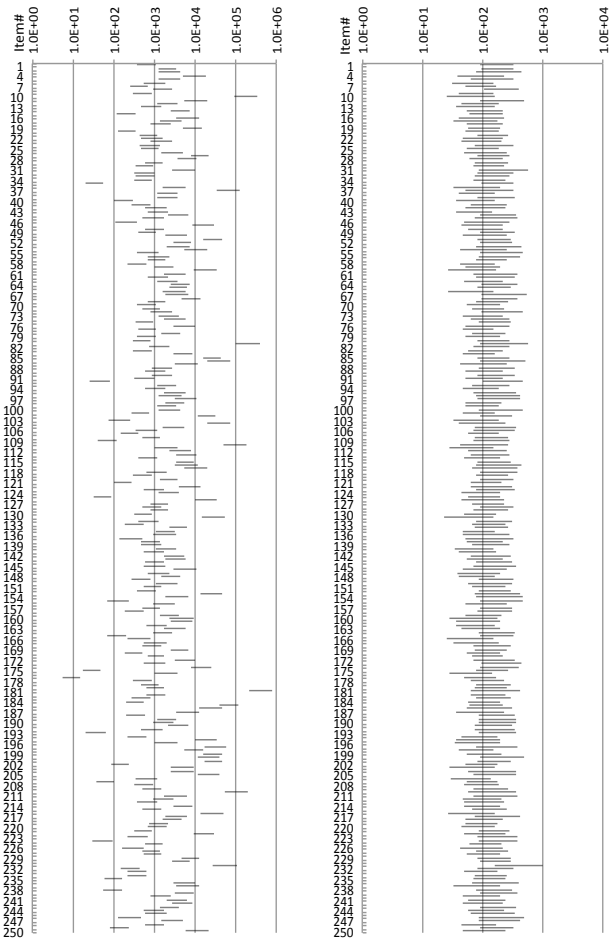
5. HARD INSTANCES FOR THE (1+1) EA

We now carry out theoretical and experimental investigations for the classical (1+1) EA on instances of the nonlinear knapsack problem. Our goal is to provide additional insights on which instances are hard to be solved by simple evolutionary algorithms. The (1+1) EA is a standard algorithm investigated in the area of runtime analysis of evolutionary computation [1, 12], and therefore well suited for understanding of working behaviour of evolutionary computing techniques for NKP. The description of the (1+1) EA considered in this section is given in Algorithm 2. It starts with a solution chosen uniformly at random and uses mutation flipping each bit with probability $1/m$ in order to produce an offspring. Being not worse than it's parent according to the fitness function, the offspring replaces it.

5.1 Theoretically Constructed Instance

In this section, we present an NKP problem instance where $(1+1)$ EA fails with a constant probability. The instance is motivated by an instance for the classical knapsack problem for which it has been shown that the (1+1) EA has an exponential expected optimisation time [17]. Furthermore, it has been used for investigations of utility functions in the area of evolutionary multi-objective optimisation [11].

Our instance consists of two cities only. The distance between the cities is set to 1. The first city contains $m = r+1$ items while the second city is a destination point free of items. We give r items of the first city the same profit and weight such that cost $c_k = p_{1k} = w_{1k} = 1$, $1 \leq k \leq r$. Similarly, item $m = r+1$ gets its profit and weight as cost $c_m = p_{1m} = w_{1m} = r+1$. We call the former items *simple* and the later one *unique*. To establish the unconstrained settings, we specify $W = 2r+1$. Subsequently, we set $v_{max} = 2$



(a) Initial Instance (b) Final Instance of EA

Figure 3: Comparison between Initial Instance and Accumulated Instance by EA

and $v_{min} = 1$ which implies $\nu = 1/W$. Finally, we define $R^* = W \cdot (2 - (r+1)/W)^2$.

In order to show why the (1+1) EA fails to find the optimal solution in polynomial time, we use arguments similar to ones discussed in [11, 17]. The basic idea is that with probability 1/2 the item $m+1$ is not included into the population. Furthermore, the expected number of simple items in the initial solution is $r/2$ and at least $(1/2 - \epsilon)r$, $\epsilon > 0$ a constant, with probability $1 - e^{-\Omega(m)}$ using Chernoff bounds. The waiting time for such a step is exponentially small in the number of bits that have to be flipping. This implies an exponential optimization time of the (1+1) EA when starting with an initial solution that has not chosen item $r+1$ and many simple items.

We relate the previous ideas to our instance of NKP. Taking into account the given properties, we now specify the objective function (1) as

$$f_{R^*}(w) = w - \frac{R^*}{2 - w/W}, \tag{12}$$

where the input $w = \sum_{k=1}^{r+1} c_k x_k$ is restricted to discrete integer values respecting the decision vector $X \in \{0,1\}^m$. When defined on the interval $[0, W]$, $f_{R^*}(w)$ reaches its unique maximum in the point $w^* = W \cdot (2 - \sqrt{R^*/W}) = r+1$.

Algorithm 2 (1+1) EA

```
1: procedure INITIALISATION
2:     Randomise m bits in the decision vector X.
3: end procedure
4: repeat
5:     procedure MUTATION
6:         for each bit in X do
7:             Randomise b ∈ [0, 1]
8:             if b ≤ 1/m then
9:                 Flip the bit
10:            end if
11:        end for
12:    end procedure
13:    procedure SELECTION
14:        Select the instance with better result
15:    end procedure
16: until Reach the maximal iteration
```

In our particular settings, achieving $w^* = r + 1$ is possible when the unique item is solely selected in the packing plan X. In other words, selecting item $r + 1$ only results in the unique global optima for the given class of instances.

For a given real $\epsilon \geq 0$, we can establish that $f_{R^*}(w) > f_{R^*}(w + r + 1)$ holds for $w \geq (1/2 - \epsilon)\, r$ starting with a certain value of r, $r \to \infty$. This implies that flipping the bit corresponding to item $r + 1$ is not accepted. To give a formal proof for the (1+1) EA we would need to consider multiple bit flips in addition. This can be done by a more detailed analysis.

5.2 Evolving Hard Instances

We use another Evolutionary Algorithm to find hard instances for given renting rates. The general setting of this EA is identical to the λEA in Algorithm 1. However, we need to evolve hard instances directly in this study instead of finding instances with high non-trivial item ratio. We therefore rewrite the fitness function of the λEA to make it able to measure the hardness of an instance directly.

A straightforward approach of measuring hardness is to record running time of the exact MIP solver as introduced by Polyakovskiy et al [14]. However, such approach can only reflect the performance of the MIP solver. As our goal is to provide additional insights on which instances are hard to be solved by simple evolutionary algorithms, we introduce *failing ratio* δ.

The failing ratio δ is defined to measure the probability of an approximation solver failing to obtain a result that is better than a preset goal. In our case, the (1+1)-EA solver listed in Algorithm 2 is considered as the approximation solver. We run it 20 times with maximal iteration 20,000. For each time, we count if the result is worse than the result from the exact MIP solver. And then we calculate the failing ratio according to the fraction of the counted number on the total tried times. For our instances, the failing ratio is simply calculated by $\delta = \frac{no.\ of\ worse\ results}{20}$.

We also run the EA based on the given values of R defined in Section 4, so that the results of the EA and λEA on the same settings of R can be compared. In detail, assuming with a value of R, the result of the λEA reported that high λ could be found and the EA indeed found the hard instances, then our conjecture of the association of the non-trivial item ratio λ and the hardness of instances can be confirmed, vice versa. This would also confirm the feasibility of our pre-processor of eliminating easy instances.

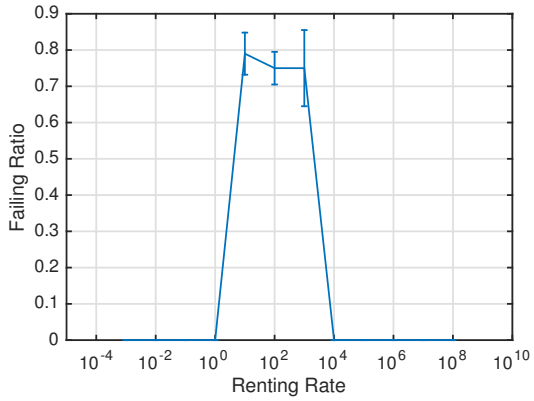


Figure 4: Failing Ratio of (1+1)-EA

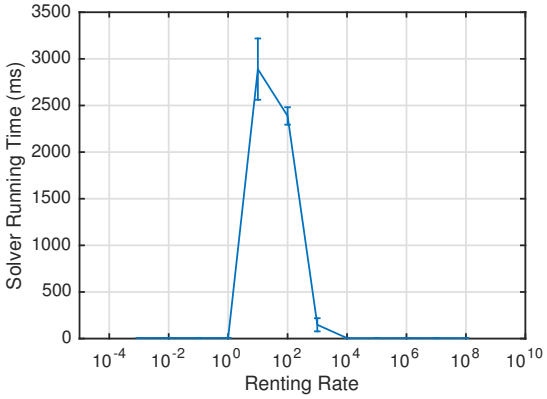


Figure 5: Runtime (ms) of the MIP Solver

In practice, we run 5 times the EA for each given R values respectively with a unified maximal iteration 10,000 to record averages and standard deviations for the failing ratio δ and runtime of both the MIP solver and the (1+1)-EA solver. The results are plotted in Figure 4, 5 and 6.

In the figures, the horizontal axises are identical to each other and to the counter axis in the figure of λEA (Figure 2). There are 12 error bars in each figure, representing 12 results according to given values of R in $\{1.0\text{e}{-3}, ..., 1.0\text{e}{+8}\}$.

In Figure 4, we observe only 3 out of 12 R values have failing ratio greater than zero, which are 1.0e1, 1.0e2 and 1.0e3. This exactly matches the results of λEA in Section 4, where instances with the λ higher than a threshold 0.8 can only be discovered when R is in such values. Moreover, for the values of R where λ is below 0.8, no hard instance can be found at all. Additionally in Figure 5, the time used to solve instances are significantly increased when R is set to the same values, which also implies that hard instances can only be found when R is in such range. However, the running time of the (1+1)-EA solver seems not to be influenced by the hardness of instance.

The results of Figure 4, 5 confirm our conjecture that with setting a threshold, the non-trivial item ratio λ could be utilised as a pre-processor to eliminate easy instances. In practice, we need to calculate the non-trivial range of

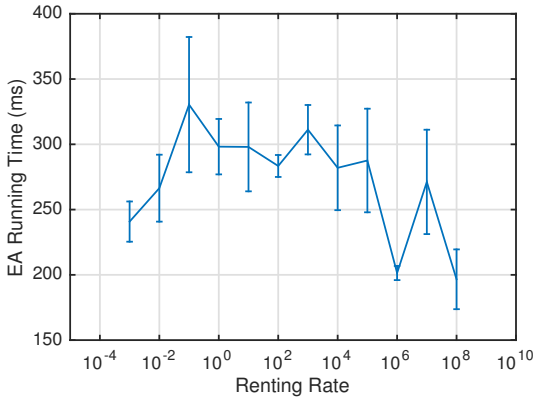


Figure 6: Runtime (ms) of the (1+1)-EA solver

R beforehand. And then we use the λEA to explore the non-trivial range of R logarithmically to find highest λ for each given R. At last the threshold could be obtained from analysing the results. By calculating λ of instances and utilising the threshold accordingly, easy instances can be eliminated efficiently.

6. CONCLUSION

With this paper, we contributed to the understanding of multi-component combinatorial optimisation problems that arise frequently in real-world applications such as supply chain management. We investigated the non-linear knapsack problem motivated by the recently introduced Traveling Thief Problem and studied the impact on the renting rate which connects the profit and the cost part of the problem. We have shown by theoretical and experimental investigations how the renting rate affects the number of items that can be tackled by simple pre-processing algorithms. Furthermore, we have constructed instances in a theoretical and experimental way where a simple baseline (1+1) EA fails to obtain an optimal solution.

Acknowledgements

The authors were supported by Australian Research Council grants DP130104395 and DP140103400.

References

[1] A. Auger and B. Doerr. *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. World Scientific Publishing Co., Inc., 2011.

[2] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Appl. Soft Comput.*, 11(6):4135–4151, 2011.

[3] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, 2003.

[4] M. Bonyadi, Z. Michalewicz, and L. Barone. 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, June 2013.

[5] M. R. Bonyadi and Z. Michalewicz. Evolutionary computation for real-world problems. In S. Matwin and J. Mielniczuk, editors, *Challenges in Computational Statistics and Data Mining*, volume 605 of *Studies in Computational Intelligence*, pages 1–24. Springer, 2016.

[6] G. Dick, W. N. Browne, P. A. Whigham, M. Zhang, L. T. Bui, H. Ishibuchi, Y. Jin, X. Li, Y. Shi, P. Singh, K. C. Tan, and K. Tang, editors. *Simulated Evolution and Learning - 10th International Conference, SEAL 2014, Dunedin, New Zealand, December 15-18, 2014. Proceedings*, volume 8886 of *Lecture Notes in Computer Science*. Springer, 2014.

[7] H. Faulkner, S. Polyakovskiy, T. Schultz, and M. Wagner. Approximate approaches to the traveling thief problem. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 385–392, New York, NY, USA, 2015. ACM.

[8] Y. Mei, X. Li, F. Salim, and X. Yao. Heuristic evolution with genetic programming for traveling thief problem. In *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*, pages 2753–2760. IEEE, 2015.

[9] Y. Mei, X. Li, and X. Yao. Improving efficiency of heuristics for the large scale traveling thief problem. In Dick et al. [6], pages 631–643.

[10] Y. Mei, X. Li, and X. Yao. On investigation of interdependence between sub-problems of the travelling thief problem. *Soft Comput.*, 20(1):157–172, 2016.

[11] F. Neumann and A. Q. Nguyen. On the impact of utility functions in interactive evolutionary multi-objective optimization. In Dick et al. [6], pages 419–430.

[12] F. Neumann and C. Witt. *Bioinspired Computation in Combinatorial Optimization:Algorithms and Their Computational Complexity*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[13] S. Polyakovskiy, M. R. Bonyadi, M. Wagner, Z. Michalewicz, and F. Neumann. A comprehensive benchmark set and heuristics for the traveling thief problem. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 477–484, New York, NY, USA, 2014. ACM.

[14] S. Polyakovskiy and F. Neumann. Packing while traveling: Mixed integer programming for a class of nonlinear knapsack problems. In L. Michel, editor, *Integration of AI and OR Techniques in Constraint Programming*, volume 9075 of *Lecture Notes in Computer Science*, pages 332–346. Springer International Publishing, 2015.

[15] G. Reinelt. TSPLIB- a traveling salesman problem library. *ORSA Journal of Computing*, 3(4):376–384, 1991.

[16] J. Stolk, I. Mann, A. Mohais, and Z. Michalewicz. Combining vehicle routing and packing for optimal delivery schedules of water tanks. *OR Insight*, 26(3):167–190, 2013.

[17] Y. Zhou and J. He. A runtime analysis of evolutionary algorithms for constrained optimization problems. *IEEE Trans. Evolutionary Computation*, 11(5):608–619, 2007.