2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

The travelling thief problem: the first step in the transition from theoretical problems to realistic problems

Mohammad Reza Bonyadi
School of Computer Science
The University of Adelaide, Adelaide,
mrbonyadi@cs.adelaide.edu.au

Zbigniew Michalewicz*
SolveIT Software
99 Frome St, Adelaide, SA 5000
zm@solveitsoftware.com

Luigi Barone
SolveIT Software
99 Frome St, Adelaide, SA 5000
lbarone@solveitsoftware.com

***Abstract*—There are some questions concerning the applicability of meta-heuristic methods for real-world problems; further, some researchers claim there is a growing gap between research and practice in this area. The reason is that the complexity of real-world problems is growing very fast (e.g. due to globalisation), while researchers experiment with benchmark problems that are fundamentally the same as those of 50 years ago. Thus there is a need for a new class of benchmark problems that reflect the characteristics of real-world problems. In this paper, two main characteristics of real-world problems are introduced: combination and interdependence. We argue that real-world problems usually consist of two or more sub-problems that are interdependent (to each other). This interdependence is responsible for the complexity of the real-world problems, while the type of complexity in current benchmark problems is missing. A new problem, called the travelling thief problem, is introduced; it is a combination of two well-known problems, the knapsack problem and the travelling salesman problem. Some parameters which are responsible for the interdependence of these two sub-problems are defined. Two sets of parameters are introduced that result in generating two instances of the travelling thief problem. The complexities that are raised by interdependences for these two instances are discussed in detail. Finally, a procedure for generating these two instances is given.**

Keywords—Real-world problems; NP-hard problems; combinatorial benchmark problems; complexity

I. INTRODUCTION

Over the past 50 years, many optimization problems have been studied by researchers from different fields of study: applied mathematics, operations research, meta-heuristics, and artificial intelligence. Most of the studied problems belong to the class of problems known as NP-hard, so that solving "large" instances of these problems to optimality is not possible [1]. These problems include the travelling salesman problem, knapsack problems, the vehicle routing problem, the multiprocessor task scheduling problem, job shop scheduling, graph colouring, clique covering, maximum flow networks, and many others. Most of these problems represented some real-world industrial environments so that solving them to optimality was important. Many *exact* methods for NP-hard problems have been proposed, e.g. branch and bound, dynamic programming, and memorization. Although such methods gained some successes in solving these problems, two main shortcomings remained: 1) none of these methods was time efficient when the problem instance was large, and 2) each application of such methods had to be tailored to the particular problem. Due to the relevance of this class of problems to real-world environments, many attempts were made to address these two shortcomings. Many *heuristic* methods emerged that offered near optimal solutions (rather than the exact optimal solutions) and these methods were much faster than exact methods. Examples include the Lin-Kernighan method for the travelling salesman problem (TSP) [2] or Pirkul's method for the multiple knapsack problem (MKP) [3]. Despite some success of these heuristic methods in addressing the first shortcoming of the exact methods (time efficiency), they were not sufficiently effective as they still suffered from the second shortcoming – they were still too problem specific. Thus, there was a need for a new class of heuristic methods that would be more generic – capable of solving a set of different problems with only minor modifications. Consequently, several *meta-heuristic* methods [4] (many of which were inspired by nature) emerged; examples include tabu search, simulated annealing, genetic algorithms, ant colony, and evolutionary algorithms. The race to design new nature-inspired and generic methods has remained an active research area for the past 30 years and many additional meta-heuristics have been proposed and applied to the NP-hard problems. All these meta-heuristic methods are *iterative* [5] – they start from constructing one or more complete solutions and gradually improve these solutions during some iterative process. Thus it is possible to track the quality of the solutions found at each iteration, so the user can stop the algorithm as soon as acceptable results are identified. This ability is rarely available in the previous methods like branch and bound or dynamic programming. Many other concepts were introduced afterwards that were connected to meta-heuristics, for example hyper-heuristics [23], and cooperation coevolution [21]. As more and more methods were proposed by researchers, the need for running systematic comparisons (evaluations) among these methods became harder and harder. Hence, to compare the efficiency

* Z. Michalewicz is also with the School of Computer Science, the University of Adelaide, Adelaide, SA 5005, Australia, and also at the Institute of Computer Science, Polish Academy of Sciences, ul. Ordona 21, 01-237 Warsaw, Poland, at the Polish-Japanese Institute of Information Technology, ul. Koszykowa 86, 02-008 Warsaw, Poland, and at SolveIT Software, level 1, 99 Frome Street, Adelaide, SA 5000, Australia.

of various methods in solving NP-hard problems many standard benchmarks (such as OR-library [6] or TSPLIB [7]) were generated, so comparisons became easier (and more meaningful). Such comparisons indicated that many meta-heuristic methods could solve many large instances of NP-hard problems to near optimality (even to optimality in some cases) in an acceptable time.

However, some researchers believe that there is a growing gap between research and practice in meta-heuristic methods, which makes these methods ineffective in solving the real-world problems [5]. The reason is that the main focus of many researchers over the years remained the same: to provide effective meta-heuristics for solving well-known benchmark NP-hard problems. However, these benchmark problems *do not* reflect the main characteristics of the real-world problems, so the fact that the meta-heuristic methods perform well on these benchmarks does not mean that they are necessarily effective in the instances of real-world problems. Some researchers tried to identify the main sources of complexity for the real-world problems. For example, in [8] the authors identified several reasons that explain why optimization problems are hard to solve. The reasons discussed in that paper included premature convergence, ruggedness, causality, deceptiveness, neutrality, epistasis, robustness, overfitting, and oversimplification. It seems that these reasons are either related to the landscape of the problem (such as ruggedness and deceptiveness) or the optimizer (like premature convergence and robustness) and they are not focusing on the nature of the problem. In a book on modern heuristic methods [9], five main reasons behind the hardness of real-world problems were discussed: the size of the problem, modelling issues, noise, constraints, and some psychological issues.

In this paper we claim that one of the main differences between the benchmark problems and real-world problems lies in the definition of *complexity*. Normally, complexity in benchmark problems refers to the scale of the problems. As an example, in 1954, a particular method for solving the TSP was introduced [10], having been tested on instances of up to 50 cities. Today, the same problem remains the focus of many researchers, with the only difference being that the size of the problem instances has been changed from 50 to 1,000 [11]. Another example is MKP, which was already of interest to researchers in 1965 [12]. As with the TSP, the problem is still one of the active domains in the field [13] with many standard benchmarks – instances include cases up to 2,500 items and 200 constraints [6, 14]. However, the complexity of real-world problems is not only limited to the size of the problem. Let us illustrate this point by the following two examples.

The first example is a real-world problem of optimizing the transportation of water tanks [15]. A company produces water tanks and sells them to some customers (we refer to their locations as stations). There is a plant (possibly several plants) that produces water tanks (different sizes) and trucks (possibly with trailers) are used to carry these water tanks to the stations for delivery. Because the tanks are empty and of different sizes, we may consider bundling (packing) the tanks inside each other. The advantage is that a truck can carry more tanks, but the price to pay for that gain is in unbundling activity. The bundled tanks are unbundled at special sites (called bases), before their delivery to customers. When the tanks are unbundled in a base, only some of them fit and are carried by the truck, as they require more space. The remaining tanks are kept in the base until the truck gets back and loads them again to continue the delivery process. The truck goes to other stations, delivers the tanks, stops at some base (if necessary) to unbundle some tanks, takes some of them and delivers them. The truck repeats this activity until all tanks are delivered. The aim of the optimizer is to select a subset of tanks (to be loaded in the truck possibly with bundling) for delivery (to corresponding costumers of course), and to determine an exact route (order of stations) that includes also all bases used for unbundling activities – to maximize the total "value" of the delivery. This total value is proportional to the ratio between the total price of delivered tanks to the total distance that the truck travels. Clearly, there are at least two sub-problems presented in the transportation of water tanks problem: (1) selecting the tanks for delivery/bundling and (2) delivering them to customers (this includes selection of appropriate bases for unbundling). Note that if the selection of the tanks is solved to optimality (the best subset of tanks is selected in a way that the price of the tanks for delivery is maximized), there is no guarantee that the overall optimum solution will be found (even if the delivery tour is solved to optimality), because the process of selecting tanks does not take into account the location of stations and bases (e.g. there might be a station that needs a high dollar value tank but it is very far from the base). Thus, the total distance of the travel is not acceptable (despite the fact it is the best for the selected tanks). On the other hand, it is impossible to select the best bases and the best order of stations before selecting tanks – without selection of tanks, the best solution (lowest possible tour distance) is to deliver nothing. Thus, solving each sub-problem in isolation does not necessarily lead us to the overall optimal solution. Note also that in the real-world case [15], there are many additional sub-problems such as scheduling of drivers, traffic on the roads and maintenance of the trucks that are interdependent.

Another real-world example is PCB (Printed Circuit Board) design in VLSI (Very Large Scale Integrated) circuits. It is well-known that one of the applications of the TSP is in PCB design (many TSP benchmarks have been named after some known PCBs [7]). In PCB design, the aim is to connect specific pins of electronic components using some tracks (wires) in such a way that summing up the length of all tracks is minimized. However, this is just one part of the problem. The complete version of the problem is as follows: place some given electronic components on a board (this part is called *placement*), with the area of the board and each component being given, and connect specific (given) pins of these components in a way that the amount of used tracks is minimized (this part is called *wiring*). In PCB design, it is sometimes the case that the tracks cannot cross the components (although there are some exceptions). This might

be for many reasons; for example, the frequency of the signals in the tracks might affect the functionality of the components; the proximity of the high voltage tracks which have significant effect on each other and should not be close to other components or even each other. . Thus, the placement of the components is very important. Accordingly, in the real-world PCBs, the challenge is not only finding the best plan for connecting pins, but also the best placement of components. Note that a solution for one part (placement or wiring) affects the solution for the other part. Hence, there are two interconnected problems in PCB design: placement and wiring, but there is one objective, that is, the lowest usage of tracks is achieved. Note that in the real-world complete version, there are many other aspects such as the different sizes of the tracks, the different sizes of the pins, and the possibility of some boards having multi-layers.

Two important characteristics can be extracted from these two real-world problems:

- *Combination* – the real-world problems usually consist of two or more sub-problems that are "combined together", and
- *Interdependence* – in real-world problems these sub-problems are interdependent in the sense that a solution for one sub-problem influences the quality of the solutions for other sub-problems.

Based on the second characteristic (interdependence), solving one sub-problem (even to optimality) in isolation is not useful without considering other sub-problems, as the best plan for one sub-problem affects the best plan for the other sub-problems. As we already discussed, the best selection of the water tanks may result in a high penalty to the delivery distance, or the placement of components in different ways may change the best wiring.

Many studies indicate [16, 17] that many real-world problems (e.g. supply chains) are modelled by interdependent sub-problems. However, this aspect of interdependence is missing in the current benchmark of optimization problems. In fact, current benchmark problems are single or multi-objective NP-hard problems with a variety of sizes, from small (TSP with a few cities, MKP with a few items) to very large (TSP with 10,000 cities, MKP with up to 2,500 items), while there are many complex problems in the real world where complexity results from the interdependence between their components rather than the number of objectives or the size of the problem. Thus, it seems that the benchmark problems are not comprehensive enough to cover current real-world problems/challenges. Hence, it is obvious that while the benchmark problems remain fundamentally somewhat dissimilar to real-world problems, the gap between optimization methods and their real-world applications is not bridged.

The main contribution of this paper is the definition of a new benchmark problem (called the travelling thief problem) that is a better approximation to real-world problems. This benchmark problem contains both introduced characteristics (combination of sub-problems and their interdependence)[2]. Further, some parameters are defined to generate two different types of interdependency between the sub-problems in the proposed benchmark problem.

II. TRAVELLING THIEF PROBLEM

In this section, a new problem called the travelling thief problem (TTP) is introduced. The problem is a combination of two well-known optimization problems (TSP and KP). Two models of TTP are introduced and their characteristics are investigated. These two models are different from each other based on the way in which the sub-problems are interdependent. It is shown that solving each sub-problem in isolation is not effective and the two sub-problems have to be considered together.

A. Travelling salesman problem

The travelling salesman problem is one of the classic NP-hard optimization problems. In this problem, there are n cities and the distances between the cities are given by a distance matrix $D=\{d_{ij}\}$ (d_{ij} the distance between city i and j). There is a salesman who must visit each city exactly once and minimize the time of the complete tour. It is assumed that the speed of the salesman is constant (v_c) and he tries to minimize the time of the tour[3]. The objective function is given by:

$$f(\overline{x}) = \sum_{i=1}^{n-1} (t_{x_i,x_{i+1}}) + t_{x_n,x_1}, \overline{x} = (x_1, \dots, x_n) \quad (1)$$

where $\overline{x}$ represents a tour, which contains all of the cities exactly once, $t_{x_i,x_{i+1}}$ is the time of the travel between x_i and x_{i+1} and it is calculated by

$$t_{x_i,x_{i+1}} = \frac{d_{x_i,x_{i+1}}}{v_c} \quad (2)$$

Clearly, f is the total time of the tour. The aim is to find x which minimizes f.

B. Knapsack problem

The knapsack problem is another NP-hard optimization problem. In this problem, there are m items $I_1, \dots, I_m$, each of which has a value (p_i) and a weight (w_i). Also, there is a thief who wants to fill his knapsack by taking some of these items. The capacity of the knapsack is limited (W). The aim of the thief is to pick the items to maximize their total value while their total weight does not exceed the knapsack capacity. The problem is modelled as follows:

$$maximize\ g(\overline{y}) = \sum_{i=1}^{m} p_i y_i, \overline{y} = (y_1, \dots, y_m) \quad (3)$$

[2] Note that, there are some other existing benchmark problems (such as Capacitated Vehicle routing problem) that are a combination of two other problems, however, the interdependency between the sub-problems in these benchmarks is fixed and also has not been investigated in detail.

[3] Note that when speed is constant, we can alternatively consider minimization of the total distance rather than the total time of the tour.

$$subject\ to \sum_{i=1}^{m} w_i y_i \le W \quad (4)$$

where $y_i \in \{0,1\}$ and $y_i = 1$ indicates picking item i and $g(y)$ is the total value of the picked items. The aim is to maximize g in a way that the total weight of the items does not exceed the capacity of the knapsack.

C. Travelling thief problem: definition

There are n cities, and the distance matrix $D=\{d_{ij}\}$ is given. Also, there are m items each of them having a value p_k and weight w_k. There is a thief who is going to visit these cities exactly once and pick some items from the cities and fill his knapsack. The maximum weight for the knapsack is W. The aim is to find a tour that visits all of the cities exactly once and gets back to the starting city, optimizing objective function(s) while the total weight of the knapsack is not violated. Note that the objective function(s) might be related to the time of the travel and/or the total value the thief gains from picking the items. In the rest of this paper, it is assumed that the thief starts from the first city. Also, the thief can pick the items from the first city only at the beginning. The problem is represented by the following parameters:

- TSP sub-problem
  - Parameters
    - Number of cities: n
    - d_{ij} is the distance between city i and j
    - Velocity v_c
  - Solution:
    - The solution for the travel is called a *tour* ($\overline{x} = (x_1, \dots, x_n)$) where x_i is a city.
- KP sub-problem
  - Parameters
    - Number of items: m
    - Weight for each item: w_k
    - Value for each item: p_k
    - Total knapsack capacity: W
  - Solution
    - The solution of the knapsack is a binary vector called *picked items* ($\overline{y} = (y_1, \dots, y_m)$) which is called for items that show which item should be picked. Each element y_k is a bit (0/1) which shows picking the item or not.
- TTP
  - Parameters
    - Availability of item I_i in each city: $A_i \subseteq \{1, \dots, n\}$,
    - The other parameters of TTP depend on how the sub-problems (knapsack and tour) interdepend. Some possibilities are described below. Also, two particular instances of TTP are given and their parameters are defined in the next sub-sections
  - Solution
    - The complete solution for TTP contains two vectors, tour ($\overline{x}$) and *picking plan* ($\overline{z}$). The picking plan is shown by $\overline{z} = (z_1, \dots, z_m)$, $z_i \in \{0 \cup A_i\}$ for all i. z_i shows that from which city the item I_i should be picked. $z_i=0$ means that item i is not picked at all.
    - The solution for TTP is constructed by offering the tour x and picking plan z.

It is obvious that there are two sub-problems in TTP, finding the best tour (x) and finding the best picking plan (z). The solution for TTP includes x and z as z expresses all information in y. There are numerous ways to connect these two problems: linking the speed of the travel to the weight of the knapsack, applying a penalty to the value of the picked items over the time of travel, assigning a different value for one item in different cities, etc. We design two TTPs, called TTP_1 and TTP_2. These two TTPs are different from each other based on the way that the two sub-problems interdepend and the number of objectives.

The interdependence of sub-problems in TTP causes complexity in designing appropriate algorithms for solving the problem. As an example, if the aim is to use an evolutionary algorithm (like GA) to solve the problem, the representation of the problem is not trivial and needs investigation. Note that a different presentation of the problem results in different efficiency of the optimization method in solving the problem (see [18] for examples on the efficiency of GA on knapsack problems with different presentations). There are at least two different approaches to applying the evolutionary algorithm to the problem. The first approach is to combine the input variables (x and z). In this case, the matter of hardness concerns what is the best way to combine them. Note that the space of x and z is completely different (x is a permutation of cities and z is a picking plan). After the combination of parameters, the designation of operators for the specific coding is another potential problem. The second approach is to design and apply cooperation coevolution to the sub-problems. In this case, setting the parameters of the method, for example. choosing a partner representative from populations, freezing time, and parameters of different species, is not an easy task [19-21].

Also, by combining these two problems, the constraints of one sub-problem influence the feasibility region of the solutions for the other sub-problem. As an example, in the original TSP, all permutations of the cities were feasible. However, in TTP, if the solution for KP becomes infeasible, the whole solution and permutations for the tour become infeasible as well. Moreover, because of their interdependence, the quality of the solution of each sub-problem is affected by the solution of the other sub-problem. This is shown by examples in each TTP model separately.

III. MODEL 1: TTP_1

In TTP_1, there is one objective, namely maximizing the total benefit. In order to make two sub-problems interdependent, two new parameters are introduced.

a) The speed of the travel (v_c in (2)) is related to the knapsack's current weight, ($v_c = (v_{max} - W_c \frac{v_{max}-v_{min}}{W})$) where W_c is the current weight of knapsack, v_{max} and v_{min} are the maximum and minimum velocity of the thief, and W is the maximum capacity of knapsack. Note that the time of the travel between two cities is calculated via (2). According to this formulation, the speed of the thief (v_c) decreases when the weight of the knapsack increases. Note that the thief travels in v_{max} when the knapsack is empty (W_c=0) and in v_{min} when the knapsack is full (W_c=W),

b) The thief has rented the knapsack and he has to pay the rent. The knapsack's rent rate is $R per time unit.

The aim of TTP_1 is to maximize the following function:

$$G(x,z) = g(z) - R * f(x,z) \quad (5)$$

where g is the total value of all picked items, R is the rent per time unit, and f is the total time of the tour. x and z are the tour and the picking plan (the solutions of TSP and KP sub-problems), respectively. Note that the travel time and weight of the picked items have been connected through the changes of the speed of travel. Thus, the function f needs to get both x and z as inputs to calculate the total time. Also, the total value of the picked items has been connected to the travel time by the rent rate.

To calculate the search space of TTP_1, we need to pick item i from city j or not to pick it at all. Thus, we have B_i=$Card(A_i)$+1 different possibilities for picking this item. For each different picking plan, we need the order of the cities, which has n! different possibilities. All in all, the entire search space is $O(n! \prod_{i=1}^{m} B_i)$.

An example of TTP_1 is provided as follows,

- n=4, m=5, W=3, v_{max}=1 and v_{min}=0.1
- $D = \begin{bmatrix} - & 5 & 6 & 6 \\ 5 & - & 5 & 6 \\ 6 & 5 & - & 4 \\ 6 & 6 & 4 & - \end{bmatrix}$
- The (p, w) for available items (I_i) are: $I_1 = (100, 3), I_2 = (40, 1), I_3 = (40, 1), I_4 = (20, 2), I_5 = (30, 3)$
- Availability (A_i) of each item in cities is: $A_1 = \{3\}, A_2 = \{3\}, A_3 = \{3\}, A_4 = \{2, 4\}, A_5 = \{2\}$
- R=$1 per time unit of the travel

The parameters for travel, availability of items, and (p, w) for each item have been specified in the following figure

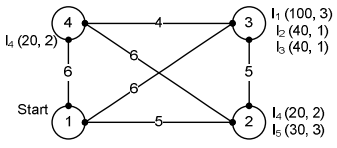

Fig. 1 a sample for TTP_1

An example solution for this TTP_1 sample is shown as follows: x = {1, 3, 2, 4} which shows the order of the cities for travel the travel, and z = {0, 3, 0, 2, 0} that shows which item is picked from which city, e.g. item I_4 is picked from city 2.

Let us calculate the objective value (5) for this particular solution. The thief starts from city 1 moving to city 3, where the distance between these cities is 6. The current weight of the knapsack (W_c) is 0, thus, v_c= v_{max} = 1, which results in $t_{1,3}$ is 6. In city 3, item I_2 is picked which results in W_c = 1. Thus v_c = 0.7 and $t_{3,2}$= 7.14. From city 2, the thief steals item I_4, which makes W_c = 3 and consequently v_c = 0.1. The thief travels to city 4 and with this velocity which takes $t_{2,4}$=60 unit of time. At the end, the thief travels back from the city 4 to the city 1 with v_c=0.1 which results in $t_{4,1}$ = 60. Putting all of these together, the tour took $f(x, z)$=60+60+7.14+6=133.14 time units. Also, the value of $g(z)$ is 40+20 = 60. Thus, the objective value is 60-1*133.14=-73.14. In this example, the optimal tour solution in isolation is x={1, 2, 3, 4} and x={1, 4, 3, 2} and the best picking plan is z={3, 0, 0, 0, 0}. The pseudo-code for calculating the objective value of TTP_1 is given as follows. The source code for the algorithm to calculate objective value for TTP_1 is available at http://cs.adelaide.edu.au/~ec/research/ttp.php.

The effect of the interdependence of TSP and KP sub-problems in TTP_1 is that by solving each sub-problem to optimality, there is no guarantee of finding the best solution for the overall problem. In fact, the solution for each sub-problem influences the objective value of the other sub-problem. That is because picking more items results in a higher total value of items, but it also results in reducing the speed. With a reduction in the speed, the rent of the knapsack increases, which influences the total benefit that the thief receives after completing the tour. To demonstrate this effect by example, Fig. 2 shows all possible solutions in the coordinate f vs. g for TTP_1 defined in Fig. 1.

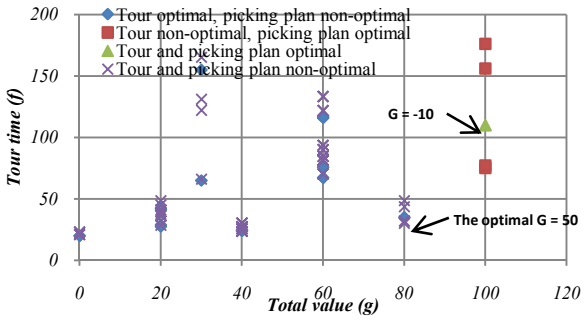

Fig. 2 Tour refers to the length of the selected tour (x) and picking plan refers to the total value of the picked items (z)

In Fig. 2, all possible solutions have been grouped into four categories:

1) First group, shown by diamonds, consists of the solutions in which the tour is optimal in isolation (without considering changes in the speed) and picking plan is not optimal,
2) Second group, shown by rectangles, consist of solutions in which the tour is not optimal and the picking plan is optimal,

3) Third group, shown by triangles, consists of the solutions in which the tour and picking plan are optimal,

4) Fourth group, shown by crosses, consists of the solutions in which both tour and picking plan are not optimal.

Fig. 2 indicates that even if the shortest tour is found and the best picking plan is guaranteed in isolation (*f* is minimized and *g* is maximized, as has been shown by triangle in the figure), the final objective value is not necessarily the optimum solution (-30 in this case while the optimum objective value is 50: *x={1, 2, 4, 3}* and *z={0, 3, 3, 0, 0}*). Note that, in the figure, the optimum solution belongs to the group of solutions for which their length and total values are not optimal choices. It seems that the two sub-problems are interdependent in such a way that solving each sub-problem does not provide any information about the optimum solution of the whole problem.

IV. MODEL 2: TTP_2

In TTP_2, there are two objectives, namely maximizing the total value while minimizing the travel time. However, three new parameters are added to interconnect two sub-problems

a) The velocity is defined as the same as inTTP_1,

b) The value of the picked items drops by time. In fact, the final value of the item at the end of the travel is not the same as its value when the thief picked the item. This value is dependent on travel time. The dropping rate is defined by $Dr^{\lceil \frac{T_i}{C} \rceil}$ where T_i is the total time that the item i is carried by the thief and C is a constant. The new value of the item i at the end of the tour is calculated by $p_i * Dr^{\lceil \frac{T_i}{C} \rceil}$ where i is the item number.

Note that the rule is to pick the items before the tour is completed. Hence, the thief cannot pick any more items when he gets back in the starting city. The aim of TTP_2 is to satisfy the following objectives:

$$G(x,z) = \begin{cases} min\, f(x,z) \\ max\, g(x,z) \end{cases} \quad (6)$$

where *x* and *z* are the tour and picking plan, respectively. The function *f* is the time of the tour which is calculated according to the distance of the cities and the weight of the picked items. Also, *g* is the total value of all picked items after completing the tour (note that the value of the items drops by the time that they were in the knapsack). The function *f* is related to the knapsack weight, in such a way that the heavier the knapsack gets, the slower the thief can travel, resulting in greater time being taken to complete the tour. Also, the value of the picked items after completing the tour depends on the time of the tour and the original value of the items. The aim is to maximize the total value of the picked items and minimize the time for completing the tour. The size of the search space for TTP_2 is calculated in exactly the same way as in TTP_1.

An example of TTP_2 is provided as follows,

- The values of *n*, *D*, *m*, *W*, v_{max}, v_{min}, and availability (A_i) of items are exactly the same as the ones in the example for TTP_1
- The pair *(p, w)* for available items (I_i) is: $I_1 = (10, 3), I_2 = (4, 1), I_3 = (4, 1), I_4 = (2, 2), I_5 = (3, 3)$
- *Dr*=\$0.9 per time unit of the travel
- *C=10*

The parameters for travel, availability of items, and *(p, w)* for each item have been replicated in the following figure

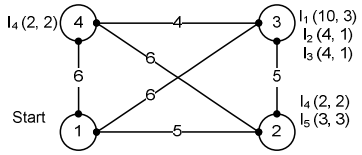

Fig. 3 a sample for TTP_2

A candidate solution for TTP_2: *x = {1, 3, 2, 4}* which shows the order of the cities for travel and *z = {0, 3, 0, 2, 0}* which shows which item is picked from which city, e.g. item I_4 is picked from city 2. Let us calculate the objective values for this particular solution for TTP_2. First, the thief starts the travel with an empty knapsack from city 1 to 3. Thus, $W_c = 0$, $v_c=1$ and consequently $t_{1,3}=6$. In the city 3, thief picks the item I_2 (this item is in the knapsack from the time 6), which results in $W_c = 1$, $v_c=0.7$, and $t_{3,2} = 7.14$. In city 3, the thief picks item I_4 (this item has been picked at the time 6+7.14=13.14), which results in $W_c = 3$, $v_c = 0.1$, and $t_{2,4} = 60$. At the end, the thief travels from the city 4 to 1 with $W_c = 3$, $v_c= 0.1$, and consequently $t_{4,1} = 60$. Thus, the total time of the tour is $f(x, z)=133.14$. Also, the total time that each item was in the knapsack was 127.14 (133.14-6=127.14) and 120 (133.14-13.14=120) for the items I_2 and I_4, respectively. Then, the total value is calculated as:

$$g(x,z) = p_2 * 0.9^{\lceil \frac{127.14}{10} \rceil} + p_4 * 0.9^{\lceil \frac{120}{10} \rceil} = 1.129 + 0.564 = 1.694$$

Thus, for this particular solution, *f* and *g* are equal to 133.14 and 1.693, respectively. The optimal solution for the sub-problems in isolation is exactly the same as the example for TTP_1. The pseudo-code for calculating the objective *G(x, z)* is given as follow. The source code for the algorithm to calculate objective value for TTP_2 is available at http://cs.adelaide.edu.au/~ec/research/ttp.php.

Note that the solution for each sub-problem in TTP_2 influences the objective value of the other sub-problem. Indeed, in TTP_2, the more items we pick, the higher value we get. However, the speed of travel decreases. This causes an increase in the travel time, which has a negative effect (values are dropped by time) on the total value of the items. Let us analyse TTP_2 in terms of the solutions of the sub-problems. Fig. 4 shows all possible solutions in the *f vs g* coordinates for the TTP_2 introduced in Fig. 3.

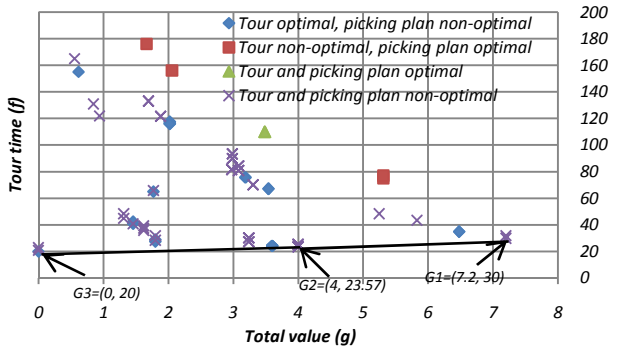

Fig. 4 Tour refers to the length of the selected tour (x) and picking plan refers to the total value of the picked items (z)

In Fig. 4, the solutions have been grouped as in Fig. 2. There are three non-dominated solutions, shown by G1 ($x=\{1, 2, 4, 3\}$, $z=\{0, 3, 3, 0, 0\}$), G2 ($x=\{1, 2, 4, 3\}$, $z=\{0, 3, 0, 0, 0\}$), and G3 ($x=\{1, 2, 3, 4\}$, $z=\{0, 0, 0, 0, 0\}$). None of these non-dominated solutions belong to groups 3 or 2. Two of them (G1 and G2) belong to group 4 and one of them (G3) belongs to group 1. Note that group 4 does not contain the best solutions in terms of the length of the tour or the total value of picked items. In G3, the length of the tour is the optimum one while the total value is 0. Thus, if we even find the optimal tour with the shortest length or the optimal picking plan for the items which maximizes the total value, there is no guarantee of finding the solutions on the Pareto front. Hence, these two sub-problems need to be considered together. Note that the solution in which its tour length and total value is optimum (indicated by triangle) is not on the Pareto front.

V. PROPOSED GENERATION PROCEDURE

In order to facilitate comparisons between different methods for solving the problem, we propose some generation methods for TTP_1 and TTP_2. First, a generation method for common parameters of both TTPs (parameters of KP and TSP) are discussed and then, generation methods for specific parameters in TTP_1 and TTP_2 are introduced separately.

TSP parameters

The first parameter for the travel is the number of cities (n). The distance matrix is considered as a symmetric matrix (the distances are the same form i to j and j to i). It is also considered that the TSPs are Euclidean. Thus, we generate the cities randomly in a plane with the height and length equal to n and then the corresponding distances are calculated.

KP parameters

The number of items m is an integer number. To generate the other knapsack parameters, the procedure in [22] is used. By using this procedure, the values of the items and their weight are correlated, making the problem harder to solve in general. In this procedure, first, the weights are generated uniformly randomly in the interval [1, 1000]. Then, the value of W is generated according to the following formula:

$$W = Tr \sum_{j=1}^{m} w_j$$

where Tr is called the tightness ratio and is a real number in the interval [0, 1]. This value controls the number of items in the final solution that can suggest the best possible solution. To generate the value of items (p_j), the following formula is used:

$$p_j = w_j + 500q_j \; for \; all \; j = \{1, \dots, m\}$$

where q_j is a uniform random number in the interval [0, 1].

TTP parameters

The parameter A_i is common between the two TTPs. To generate A_i for all i, we generate an appearance probability for each item (P_i) in the cities. The aim is to set the P_i in such a way that the better the item, the smaller the length of A_i will be (i.e. appears less frequently in the cities). We propose to generate P_i based on the quality of the items, defined as $s_i = {}^{w_i}/_{p_i}$. s_i for all items i are normalized between P_{min} and P_{max}. Then for each item i, we iterate over all cities and decide (based on P_i) if this item should appear in that city or not. Note that, in this case, the worse the item, the more probable its appearance in more cities. We propose considering $P_{min}=2/n$ and $P_{max} = 0.8$. Also, to guarantee that all items appear at least once over all cities, A_i is filled by a random city at the beginning. The v_{max} and v_{min} are considered constants equal to 1 and 0.1, respectively.

InTTP_1, the rent rate (R) should also be generated. . In order to guarantee that the best objective of TTP_1 is positive, we propose to calculate R as $R = r * \frac{E_p}{E_t}$ where r is a random number in the interval [0.05, 0.25], E_p and E_t are the estimated best total value of picked items and the estimated best achievable tour time respectively. We calculate E_p by considering picking the highest value item as much as we can. Also, because the tightness value is known, the solution contains $Tr*m$ items. Thus, E_p is calculated as

$$E_p = Tr * m * Argmax_{i=\{1\dots m\}}(p_i)$$

Also, in order to calculate E_t, we consider that the thief travels with v_{max} for the whole trip and the shortest distance within the distance matrix D is considered as the distance between all cities. Then, E_t is calculated by

$$E_t = Argmin_{i,j}(d_{ij}) * n/v_{max}$$

For TTP_2, two parameters Dr and C should be generated. The aim is to generate these two variables in a way that the most valuable item becomes worse than the least valuable one if it is selected in an early stage of the tour. It is proposed to generate Dr with a uniform random number in the interval [0.7, 0.9]. The idea is to satisfy $u * Dr^{\frac{T}{C}} = r * l$ where u and l are the most and least valuable items in the available items, respectively. Also, T is considered as the estimated average of maximum time that an item is in the knapsack. Thus, T can be considered as the E_t/v_{min} because E_t is the estimated minimum tour time and dividing that by v_{min} results in the maximum time of the tour for the shortest path. Also, r is a random value proposed to be in the interval [0.2, 0.7]. Thus, the value of C is calculated by:

$$C = \frac{ln(Dr) * E_t}{v_{min} * ln\left(\frac{rl}{u}\right)}$$

Some standard test problems for TTP_1 and TTP_2 have been generated using this procedure and they are available online at http://cs.adelaide.edu.au/~ec/research/ttp.php.

VI. CONCLUSION

In this paper, it was argued that the growing gap between the real-world and theory (specified by some other researchers before) is the result of focusing the theory on solving benchmark problems rather than real-world problems. As the benchmark problems cannot reflect all characteristics of many real-world problems, the solutions that are effective for solving benchmark problems are not necessarily effective in solving real-world problems. It was shown that usually real-world problems are a combination of at least two other sub-problems which are interdependent. This interdependency is conducted via some parameters that link the sub-problems. Using this interdependency, the solution for each sub-problem influences the quality of the other sub-problem (s). This aspect is introduced in this paper as one of the main complexities in real-world problems. A new problem (called the travelling thief problem, TTP) was introduced that was a combination of two other well-known problems, i.e. the travelling salesman problem (TSP) and the knapsack problem (KP). Two sets of parameters for interdependence between these two problems (TSP and KP) were defined resulting in two different TTPs (called TTP_1 and TTP_2). It was shown that even if the TSP and the knapsack are solved to optimality, the final solution is not necessarily the optimal solution for TTPs. The procedure for generating TTP_1 and TTP_2 was given, thereby helping researchers to compare their methods for solving these two problems. There are many other parameters and constraints that can be added to a TTP to reflect the real-world characteristics. For example, locking sub-solutions, which are frequently used in industry as some parts of the systems are prescheduled, provide one possibility. Another possibility would be un-planned maintenance on the roads (some cities/items unreachable/unavailable) according to a specific probability distribution. This constraint is very common in real-world examples where most equipment in industry has a breakdown probability distribution. Although one might not be able to find direct applications of TTP in the real world, the problem reflects one of the main complexity sources (interdependency between sub-problems) in real-world problems. As a future work, we plan to design a general model based on graphs for different classes of problems, so that the interdependency becomes easier to investigate. This general model is also useful for introducing different types of interdependencies in a more strategic way rather than simply putting different problems together to generate new benchmarks. Also, as was mentioned previously in this paper, it is unclear which approaches can be more beneficial for solving TTPs with different types of interdependencies. Hence, investigation of approaches for solving the TTP with different types of interdependencies is another direction of research.

ACKNOWLEDGMENTS

The authors would like to express their great appreciation to Maris Ozols, Frank Neumann, and Markus Wagner for their helpful discussions over the possible designations of the early types of TTP. This work was partially funded by the ARC Discovery Grants DP0985723, DP1096053, and DP130104395, as well as by grant N N519 5788038 from the Polish Ministry of Science and Higher Education (MNiSW).

REFERENCES

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*: WH Freeman & Co., 1979.
[2] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Oper. Res.,* vol. 21, pp. 498-516, 1973.
[3] H. Pirkul, "A heuristic solution procedure for the multiconstraint zero one knapsack problem," *Na. Res. Log. (NRL),* vol. 34, pp. 161-172, 1987.
[4] F. Rothlauf, *Design of Modern Heuristics: Principles and Application*: Springer, 2011.
[5] Z. Michalewicz, "Quo Vadis, Evolutionary Computation?," *Advances in Computational Intelligence,* pp. 98-121, 2012.
[6] J. E. Beasley, "OR-Library: Distributing test problems by electronic mail," *J. of the Oper. Research Society,* pp. 1069-1072, 1990.
[7] G. Reinelt, "TSPLIB—A traveling salesman problem library," *ORSA journal on computing,* vol. 3, pp. 376-384, 1991.
[8] T. Weise, M. Zapf, R. Chiong, and A. Nebro, "Why is optimization difficult?," *Nature-Inspired Algorithms for Optimisation,* pp. 1-50, 2009.
[9] Z. Michalewicz and D. B. Fogel, *How to solve it: modern heuristics*: Springer-Verlag New York Inc, 2004.
[10] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *J. of OR soc. of Am.,* pp. 393-410, 1954.
[11] C. D'Ambrosio, A. Lodi, and S. Martello, "Combinatorial traveling salesman problem algorithms," *Wiley Encyclopedia of Operations Research and Management Science,* 2011.
[12] F. Glover, "A multiphase-dual algorithm for the zero-one integer programming problem," *Operations Research,* pp. 879-919, 1965.
[13] V. Boyer, M. Elkihel, and D. El Baz, "Heuristics for the 0–1 multidimensional knapsack problem," *E. J. of OR,* vol. 199, pp. 658-664.
[14] F. Glover and G. Kochenberger, "Critical event tabu search for multidimensional knapsack problems," ed: Kluwer Academic Publishers, 1996, pp. 407–427.
[15] I. M. J. Stolk, A. Mohais, Z. Michalewicz, "Combining Vehicle Routing and Packing for Optimal Delivery Schedules of Water Tanks," *OR Insight,* 2013, doi:10.1057/ori.2013.1
[16] R. Chiong, T. Weise, and Z. Michalewicz, *Variants of evolutionary algorithms for real-world applications*: Springer, 2011.
[17] A. M. Maksud Ibrahimov, S. Schellenberg, Z. Michalewicz "Evolutionary Approaches for Supply Chain Optimisation – Part 2," *Inter. J. of Inte. Computing and Cybernetics,* vol. 5, pp. 473 – 499, 2012.
[18] J. Tavares, F. B. Pereira, and E. Costa, "Multidimensional knapsack problem: A fitness landscape analysis," *IEEE Trans. on Sys. Man and Cyber. Part B,* vol. 38, pp. 604-616, Jun 2008.
[19] M. R. Bonyadi and M. E. Moghaddam, "A Bipartite Genetic Algorithm for Multi-processor Task Scheduling," *International Journal of Parallel Programming,* vol. 37, pp. 462-487, Oct 2009.
[20] M. Ibrahimov, N. Wagner, A. Mohais, S. Schellenberg, and Z. Michalewicz, "Comparison of cooperative and classical evolutionary algorithms for global supply chain optimisation," in *CEC*, 2010, pp. 1-8.
[21] M. Potter and K. De Jong, "A cooperative coevolutionary approach to function optimization," *PPSN III,* pp. 249-257, 1994.
[22] A. a. G. P. Freville, "An Efficient Preprocessing Procedure for the Multidimensional 0-1 Knapsack Problem," *Discrete Applied Mathematics,* vol. 49, pp. 189–212, 1994.
[23] Burke, Edmund K., et al. "Hyper-heuristics: A survey of the state of the art." Technical report, NO. NOTTCS-TR-SUB-0906241418-2747 (2010), School of Computer science and onformation technology, University of Nottingham